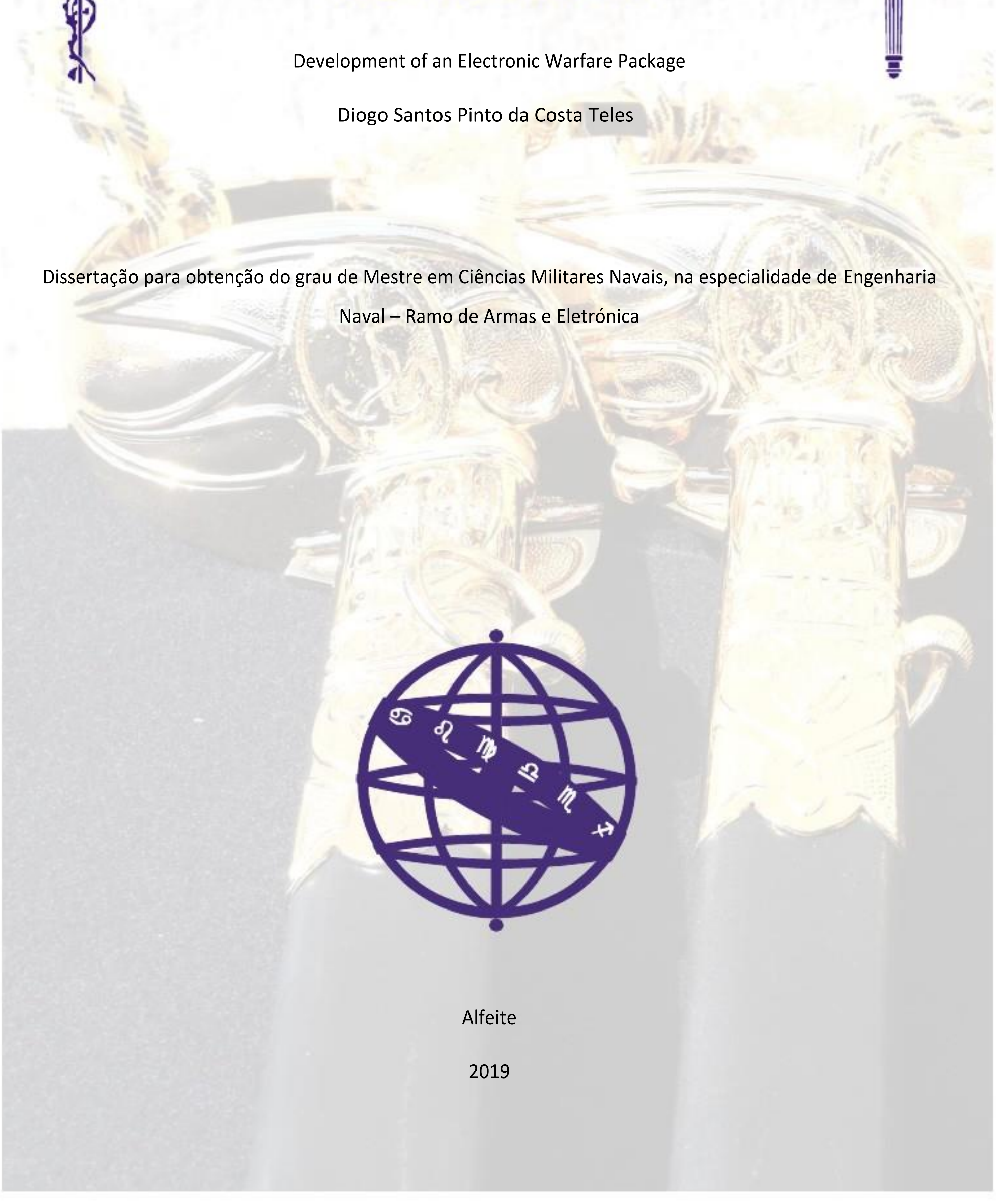


Development of an Electronic Warfare Package

Diogo Santos Pinto da Costa Teles

Dissertação para obtenção do grau de Mestre em Ciências Militares Navais, na especialidade de Engenharia Naval – Ramo de Armas e Eletrónica

Alfeite

2019

ESCOLA NAVAL

talant de bien faire

Development of an Electronic Warfare Package

Diogo Santos Pinto da Costa Teles

Dissertação para obtenção do grau de Mestre em Ciências Militares Navais, na especialidade Engenheiros Navais – Ramo de Armas e Eletrónica

Orientação de: CTEN EN-AEL Monteiro Marques

Coorientação de: Professor Doutor Vítor Lobo

O Aluno Mestrando	O Orientador
_________________________	_________________________
ASPOF EN-AEL Costa Teles	CTEN EN-AEL Monteiro Marques

Alfeite

2019

“There is much more to electronic warfare than simply detecting enemy transmissions.”

Martin Van Creveld Technology and War, 1989

Acknowledgements

I couldn’t complete this thesis without the support of several people and entities. So, I would like to thank:

- First of all, my mother and my sister that gave me the education and support that allowed me to be who I am;
- My closest friends which have been of major importance in my life, along with “Curso Jorge Álvares”;
- ASPOF Hipólito Lopes and ASPOF Rodrigues Marante with whom I’ve worked together in a group effort to improve Navy’s capacities;
- CTEN Monteiro Marques for the supervision of my thesis and all the guidance provided;
- Professor Victor Lobo for all the help and for providing the opportunity of developing my thesis in the way I did;
- 1TEN Gonçalves Capela for all the help on the software development and guidance;
- UAVision for the help and resources provided for the software and hardware development;
- CTEN Silva Ângelo and 1TEN Seixas Nunes from CITAN for all the help given that contributed actively for the realization of this thesis;
- 1TEN Mendes Lança and all CEOV team for the advices and contributions;
- CMG Farinha Alves for the help on the execution on the tests and experiments;
- All the Portuguese Marines Electronic Warfare Section for the help and support on the execution of tests and experiments and the shared knowledge;
- Crew of NRP D. Francisco de Almeida, which provided me an excellent internship and the opportunities to write this thesis;
- Crews of NRP Álvares Cabral, NRP Vasco da Gama and NRP António Enes for the support on the execution of several tests and experiments;
- TCOR Paulo Rosendo, CAP João Chora and 1SAR Joaquim Machado from the Anti-Air Artillery Regiment Nº1 and Artillery Regiment Nº5 for the help provided;
- 1TEN Ramos Maia and 1SAR Duarte Guerra from CADOP for the information provided;
- Finally, I’d like to thank to people that positively shaped my life through the Naval Academy, specially: 1TEN Anjos Fragoso, 2TEN Mendes Quina, SAJ Mauricio Sousa, Vital and my XO’s.

Abstract

Nowadays with the development and cost reduction of unmanned vehicles, armed forces around the world have been using these systems as a replacement or complement for conventional manned systems. Because of all the advantages inherent to the use of these vehicles, there have been many possible applications for these systems. One of possible applications is its use as a vehicle for carrying an Electronic Warfare package. Electronic Warfare, since the military started using and depending on the electromagnetic spectrum to achieve their objectives, has been a major warfare area of high interest.

In my thesis, I suggest the creation of an Electronic Warfare package with jamming and spoofing capacities for communication systems, designed to be implemented on an unmanned vehicle, with remote control dedicated programs. To do this, I took advantage of the Software Defined Radio technology along with the GNU Radio software, installed on a Raspberry Pi computer in order to be transportable even by small vehicles.

Keywords: Unmanned Vehicles; Electronic Warfare; Software Defined Radio; GNU Radio; Jamming; Spoofing

Resumo

Atualmente, com o desenvolvimento e a redução de custos dos veículos não tripulados, as forças armadas em todo o mundo têm utilizado estes sistemas como substituto ou complemento aos sistemas tripulados. Devido às vantagens inerentes à sua utilização, têm havido inúmeras aplicações para estes sistemas. Uma das possíveis aplicações é a sua utilização como veículo para transporte de um pacote de Guerra Eletrónica. A Guerra Eletrónica, desde que os militares começaram a utilizar e a depender do espetro eletromagnético para alcançar os seus objetivos, tem sido uma das principais áreas da guerra de elevado interesse.

Nesta dissertação, sugiro a criação de um módulo de Guerra Eletrónica com a capacidade de efetuar empastelamento e mistificação de a sistemas de comunicações, idealizado para ser implementado num veículo não tripulado, com programas dedicados para controlo remoto. Para isto, recorri à tecnologia do Radio Definido por Software assim como ao software GNU Radio, instalado num computador Raspberry Pi por forma a poder ser transportado também por pequenos veículos.

Palavras-Chave: Veículos não Tripulados; Guerra Eletrónica; Rádio Definido por Software; GNU Radio; Empastelamento; Mistificação

Contents

List of Figures

List of Tables

Acronyms

AAW – Anti-Air Warfare

AIS – Automatic Identification System

ASMD – Anti Ship Missile Defence

ASW – Anti-Submarine Warfare

ASuW – Anti-Surface Warfare

AWGN – Additive White Gaussian Noise

C2 – Command and Control

COMINT – Communications Intelligence

COTS - Commercial Off the Shelf

CTS – Clear to Send

CW – Continuous Wave

DSP – Digital Signal Processing

DSSS - Direct-sequence spread spectrum

EA – Electronic Attack

ECM – Electronic Counter Measures

ED – Electronic Deception

ELINT – Electronic Intelligence

EM – Electromagnetic

EME – Electromagnetic Environment

EP – Electronic Protection

ES – Electronic Warfare Support

ETA – Estimated Time of Arrival

EW – Electronic Warfare

GCS – Ground Control Station

GNSS – Global Navigation Satellite System

GPP - General Purpose Processors

GPS – Global Positioning System

IF – Intermediate Frequency

IMO – International Maritime Organization

IP – Internet Protocol

ISM – Industrial, Scientific and Medical

ISR – Intelligence, Surveillance and Reconnaissance

ISTAR - Intelligence, Surveillance, Target Acquisition and Reconnaissance

ITU – International Telecommunications Union

IW – Information Warfare

NATO – North Atlantic Treaty Organization

PNT – Positioning, Navigation and Timing

PSD – Power Spectral Density

RF – Radio Frequency

RTS – Request to Send

SDR – Software Defined Radio

SIGINT – Signal Intelligence

SNR – Signal to Noise Ratio

SOLAS – Safety of Life at Sea

UAS – Unmanned Aerial System

UAV – Unmanned Aerial Vehicle

US – United States

UxS – Unmanned Systems

VHF – Very High Frequency

1 Introduction

In this chapter, there will be presented the motivation, research question and hypothesis and the structure of the thesis.

1.1 Motivation

Nowadays unmanned systems (UxS) and systems are an accessible tool for almost everyone. The technological advances allow us to buy UxS for relatively low prices. This is having an impact on how military and civilian operations are being carried out [1], causing UxS to operate on more complex missions where higher levels of situational responsiveness and autonomous decision making are necessary [2]. In today's battlefields, UxS are highly integrated on the environment, bringing new intelligence, surveillance and reconnaissance (ISR) and tactical support capabilities to military forces [3], trading the risk of losing human life by systems. With this, the military around the world are exploiting the UxS potential in the various warfare areas, including Electronic Warfare (EW).

Electronic Warfare is major warfare area comprehending any action on the electromagnetic (EM) environment. Because military forces are becoming highly dependent on EM spectrum operations, they recognize its vulnerabilities and the importance on controlling this warfare environment [4]. On the last decade, there were many examples of how military forces are vulnerable to EW actions. On the 12th of April 2014, the Russian military claimed that they disabled the Aegis Combat system of the US Navy destroyer USS Donald Cook with an EW payload [5]. Also, during the North Atlantic Treaty Organization (NATO) exercise "Trident Juncture 2018", the Swedish claimed that the Russian military tried to jam the Global Positioning System (GPS) and that this wasn't the first time they did so [6].

Because EW is an area of great interest to the military around the world, including the Portuguese Navy, this thematic proved to me a big motivation to develop additional work in this area.

1.2 Research Question

As stated before, in the last years UxS became a reality on modern battlefield scenarios, changing the way military operations are carried out. Along with this, EW represents an area of great importance to military forces. Combining these two subjects in one allows the formulation of the following research question:

“Is it possible to implement an EW package on an UxS using low-cost technologies with the purpose of improving the EW capacities for the Portuguese Navy?”

To answer this question, the following hypothesis is proposed:

“By using an SDR transceiver and a Raspberry Pi computer, it is possible to build a device with jamming and spoofing capabilities on various communication systems, capable of being implemented on an UxS.”

1.3 Research Method

The research method is done according to [7] and [1]. First, a research question is formulated in order to set a goal and purpose for the thesis. Next, a research background is made were a research about the subject and related work is made. This is followed by the hypothesis formulation where is proposed a solution for the research question. The next step is the design experiment where the proposed solution in executed. Next, is the hypothesis test in which the designed experiment is tested. After, is the interpretation and analysis of the results and finally, the publishing of the results in order to share the developed work with the scientific community.

1.4 Thesis Structure

This thesis is divided in five chapters. In the first chapter, the motivation that led to the development of this work, the chosen research method and the explanation on the thesis’ structure are presented.

The second chapter is where the research of the background is done. There is an explanation on EW basics, existing jamming techniques and types, and the systems that compose the capabilities of the EW package and their importance.

In the third chapter it is presented all the necessary hardware interface a software programming necessary to create Electronic Attacks to the systems described on the second chapter.

The fourth chapter is where the tests and results made in order to prove and validate the capacities of the EW package, are presented.

Finally, in the fifth chapter, conclusions are made in accordance to the suggested hypothesis along with suggestions for future work on the theme.

2 Background

In this chapter, an introduction to EW is made, as well as an explanation of its divisions. Also, several Jamming techniques and jammer types are presented along with the identification of the bands, systems and application on naval operations where the EW package might apply, including a brief analysis of the importance and applicability of EW in the North Atlantic Treaty Organization.

Information Warfare (IW) are the warfare-like actions against any information system or the measures to protect our own systems. The IW applied military setting is Command and Control (C2) warfare which is composed by: physical destruction of information systems, psychological operations, deception, operational security and EW [8]. The latter is the focus of my thesis.

EW is defined as the military action involving the use of the EM spectrum and directed energy to control the EM spectrum or to attack the enemy [9]. In this definition, no reference is made to the equipment used but to the objective and purpose of the mission and because of this, EW is a vast concept that covers several operations.

EW can be divided by the type of operation in three categories: Electronic Attack (EA), Electronic Protection (EP) and Electronic Warfare Support (ES) [10].

According to [9] and [10]:

Electronic Attack is the division of EW involving the use of electromagnetic or directed energy to attack personnel, facilities or equipment with the intent of neutralizing or destroying the enemy's combat capability. It can be sub-divided in jamming, deception, anti-radiation missiles and directed energy weapons.

Jamming is the most common sub-division of EA. Jamming is the emission of EM energy with the objective of interfering with the enemy's use of the EM spectrum. Jamming can be divided in two divisions according to the type of signal: Radar Jamming and Communications Jamming. In EW applied to communications, the purpose of jamming is to break the enemy's communications link, denying them access to the content of a transmitted emission and, thereby, preventing them from access desirable information [11]. Jammers can have different configurations. For example, there are standoff jammers which are the ones that operate in the friendly held battlespace, stand-in jammers that operate in the hostile held battlespace, the narrowband jammers that only attack the carrier frequency and the barrage jammers that emit a broad range of frequencies simultaneously [8].

Electronic Protection is the division of electronic warfare involving actions taken to protect personnel, facilities and equipment from any effects of friendly or enemy employment of EW that degrade, neutralize or destroy friendly combat capability. This includes the employment of techniques in order to minimize the effects of enemy's or friendly's EA and ES actions [12]. The following table lists some of the available EP techniques used by modern anti-ship missiles:

Table 1 - EP Actions in Modern Anti-Ship Missiles

Missile	Country of Origin	EP Action
Harpoon Block 1C	United States of America	Frequency Agility
Harpoon Block 2	Unites States of America	Doppler Processing
Exocet MM40 Block 2	France	Frequency Agility / Leading Edge Track / Home-on-Jam
Exocet MM40 Block 3	France	Doppler Processing
RBS 15M/F/K	Sweden	Frequency Agility / Jittered PRF / Home-on-Jam
RBS 15 Mk3	Sweden	FM-CW
SS-N-22	Russia	Supersonic / Passive ARM
SS-N-25	Russia	Passive ARM / Coherent Processing
SS-N-27	Russia	Zigzag Flight Path
Otomat Mk2 Mod 4	France/Italy	Frequency Diversity

Electronic Warfare Support is the division of electronic warfare involving actions tasked by, or under direct control of, an operation commander to search for, intercept, identify and locate sources of intentional and unintentional radiated EM energy. Even though ES and signal intelligence (SIGINT) [which is includes communications intelligence (COMINT) and electronic intelligence (ELINT)] both include receiving enemy transmissions, they are differentiated in the purpose for which the transmissions are received. In COMINT and ELINT the received enemy signals (communication and noncommunication) are used to gather intelligence from the carried information in the signals and to determine details of the enemy's EM systems. On the other hand, ES is the collection of enemy signals (communication and noncommunication) with the purpose of taking and action against the signal or the associated weapon with these signals [13].

In the following figure, the three divisions of EW are presented along with the types of operations applicable to each one:

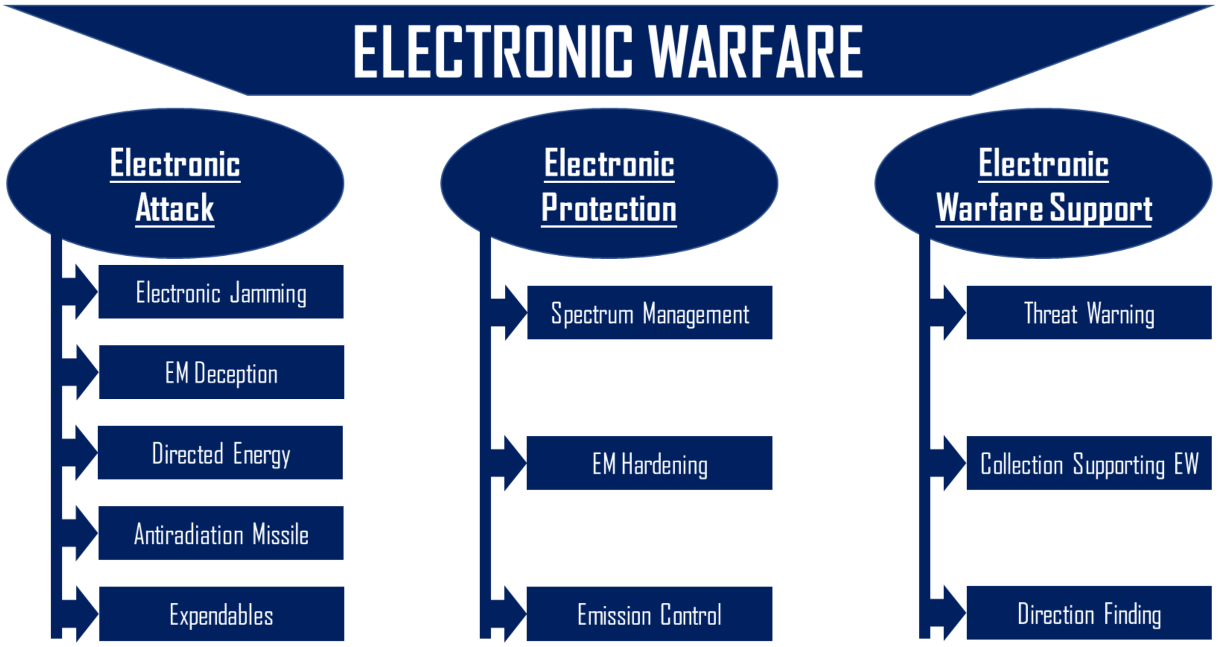


Figure 1 - Overview of EW [4]

These divisions evolved from the former definitions that were Electronic countermeasures (ECM), Electronic counter-countermeasures (ECCM) and Electronic warfare support measures (ESM).

2.1 Electronic Warfare in NATO

EW in the NATO is defined as "The military action that exploits EM energy to provide situational awareness and achieve offensive and defensive effects. EW, the conduct of EM Operations (Figure 2), is warfare in the Electromagnetic Environment (EME)" [14].

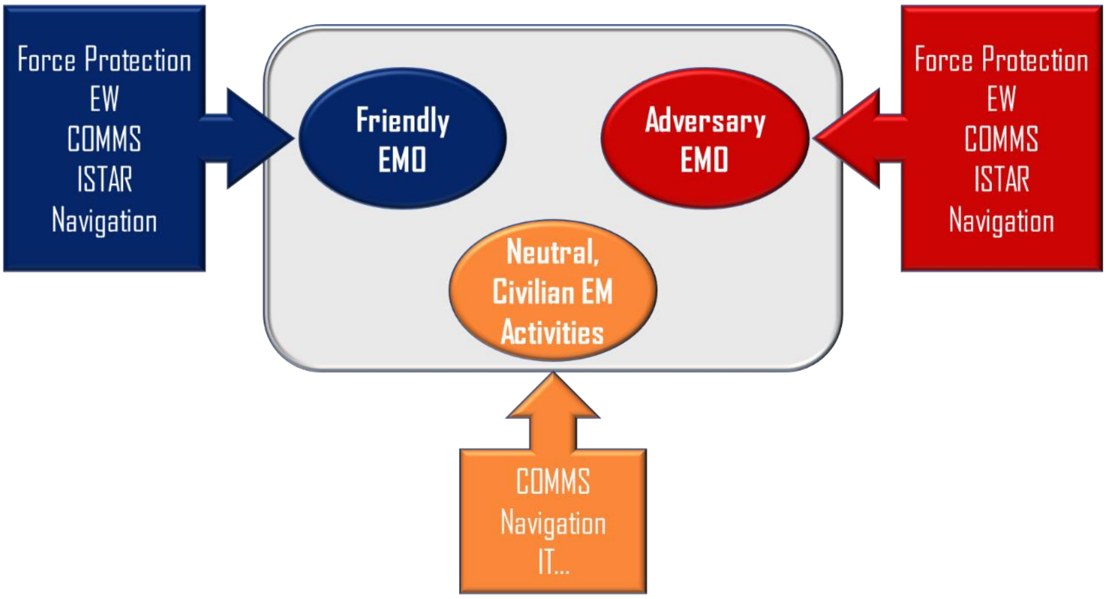


Figure 2 - EM Operations Concept [10]

NATO also acknowledges that the dependency of electronics in the modern battlefields and the threat posed to NATO by the increased EW potential of its adversaries, increase its need of controlling the EM environment by having an effective EW doctrine defined by NATO as "the best way to accomplish Joint EW operations". Specifically, EW has a crucial role in the C2 Warfare and Information Operations, being considered a central discipline of both. Another importance of EW is that the Rules of Engagement my often permit EW when other activities are inappropriate.

The uses of EW in NATO include the control of the EM spectrum to ensure provision, protection, exploitation and attack of C2 and other battlespace electronic systems; area, force or platform protection; Intelligence, Surveillance, Target Acquisition and Reconnaissance (ISTAR) and offensive and defensive operations.

There are three possible EW actions: EA, Electronic Defence and Electronic Surveillance, described in Figure 3:

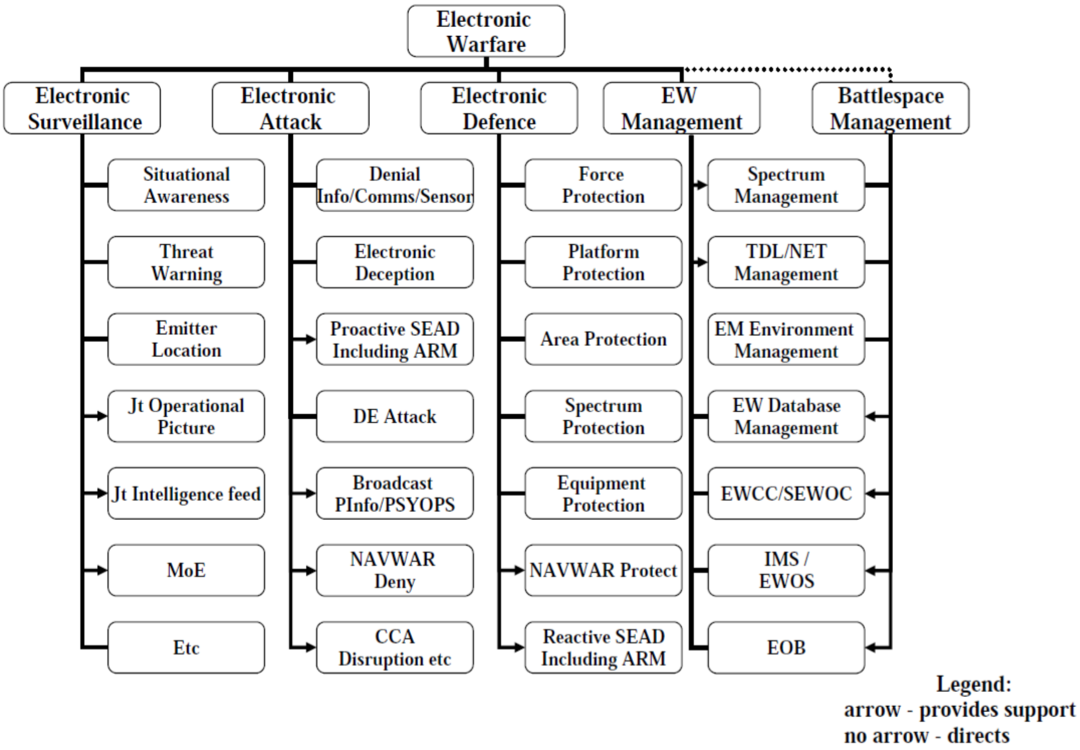


Figure 3 - EM Actions and Effects [14]

Electronic Jamming is a considered a type of operation of Electronic Counter Measures (ECM). In NATO operations, Electronic Support Measures, Electronic Counter Measures and Electronic Protective Measures are to be used as EW divisions instead of EA, EP and ES definitions used by some authors and some countries like the United States. Electronic Jamming is defined by NATO [14] as “the deliberate radiation, re-radiation or reflection of EM energy, with the objective of impairing the effectiveness of electronic devices, equipment or systems being used by an adversary. Co-ordination of jamming operations is a J3 (Current Operations) responsibility in consultation with J2 (Operational Intelligence), J6 (Communications and Information Systems) and the Electronic Warfare Coordination Cell (EWCC). Co-ordination and planning are conducted at the highest level of command, but control should be vested in the appropriate operational commander.”

Another important type of operation of ECM is Electronic Deception (ED). Electronic Deception, according to NATO [14], is “the deliberate radiation, re-radiation, alteration, absorption or reflection of EM energy in a manner intended to confuse, distract or seduce an adversary or his electronic systems.” ED can also be used to assist with military deception, principally against communications systems. Therefore, ED should be considered during the development of any deception plan. The J2 staffs provide information on the adversary's use of the EM spectrum, vulnerability, surveillance capabilities, and their likely reaction to deception, while the EW staffs provides the intelligence staff with reports indicating their actual reaction to implemented ED operations.

ED is considered effective when the adversary relies heavily on EM emissions (which may cause the adversary to react in a manned prejudicial to his interests by manipulating, distorting or faking its transmissions), when the adversary is dependent on the intercept of our EM emissions for his own intelligence collection or at a critical time in the adversary's operations.

On Figure 4 is shown the relationship between the different areas in EW and in the EM environment:

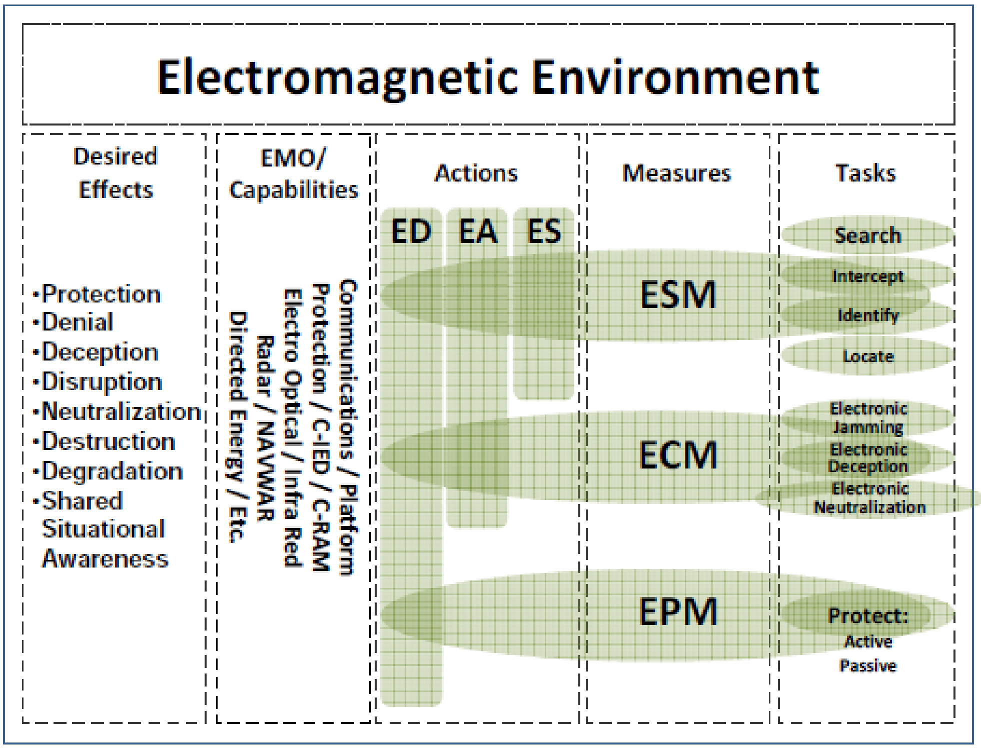


Figure 4 - Relationship between the several areas of EW [14]

For NATO, the applicability of EW in Naval operations is vast. Naval forces, acting alone or in a Joint Force use all aspects of EW in order to perform their tasks. Especially when applying the principles of EW in amphibious operations, anti-surface warfare (ASuW), anti-submarine warfare (ASW) and anti-air warfare (AAW) being included anti-ship missile defence (ASMD) in the latter. Naval EW resources are used to deny, deceive destroy or exploit the adversary’s use of the EM spectrum and to protect the use of the EM spectrum for friendly use.

2.2 Jamming Techniques

There are several techniques and strategies to make jamming. These techniques vary according to the type of system that we are trying to jam, how much power does it use, the modulation, among others. Also, the duration of the jamming in relation to the total length of the signal is important. For example, in voice communications, it has been demonstrated that jamming about 30% of the transmission is enough to degrade the message to a point where it makes it understandable [15].

In modern communications, there are also several methods of transmitting a signal making it harder to jam. These methods include frequency-hopping, direct sequence or spread spectrum where the signal is spread by the EM spectrum, resulting in a large bandwidth and hard to jam and detect signal.

Another consideration is the possibility of transmitting a signal where the jamming signal is superior to the received signal. As explained by [16] it is possible to communicate over a channel disturbed by Additive White Gaussian Noise (AWGN) and even hide a signal within the noise by using more dimensions than the ones necessary for the transmission.

2.3 Jammer Types

In order to jam a signal, there are several types of jammers that can be used. Each one of them has their advantages and disadvantages and the characteristics of the intercepted signal should be taken into consideration before choosing the type of jammer. According to different authors, there are various ways to classify jammer types.

Jack Holmes in [17] analyses five types of jammers shown on Figure 5:

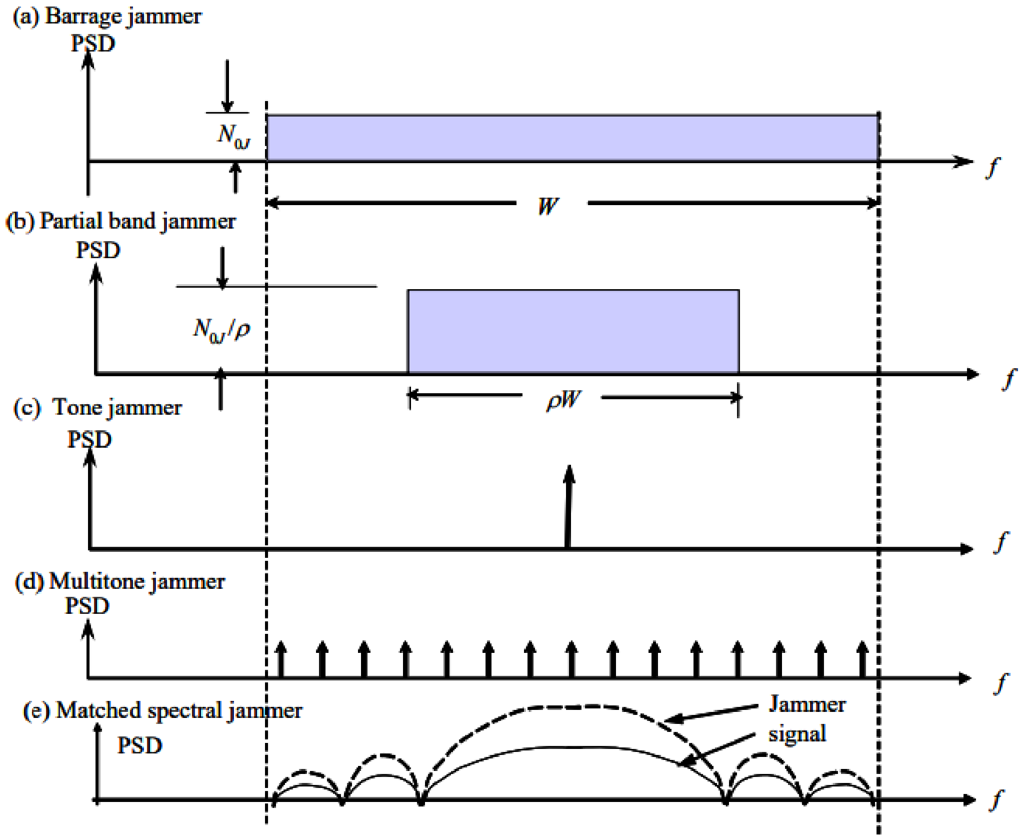


Figure 5 - Five jammer spectral densities types [17]

The first one (a), the barrage jammer, which he considers the least efficient jammer, transmits a band-limited Gaussian noise. This jammer is used to cover a large section of the signal spectrum, simulating the creation of an additional source of AWGN, in order to deny the counteraction of frequency hopping to a jammer-free zone.

However, when these jammers and the signal generators are limited on the power resource of the transmitter, they become vulnerable to spread spectrum techniques, which are the most efficient EP measure against barrage jammers. This is demonstrated in Figure 6 where a rectangular signal (left) and a spread spectrum signal (right) are jammed and, after passing through a matched filter, the spread spectrum signal is easily observable in contrary to the rectangular one [18].

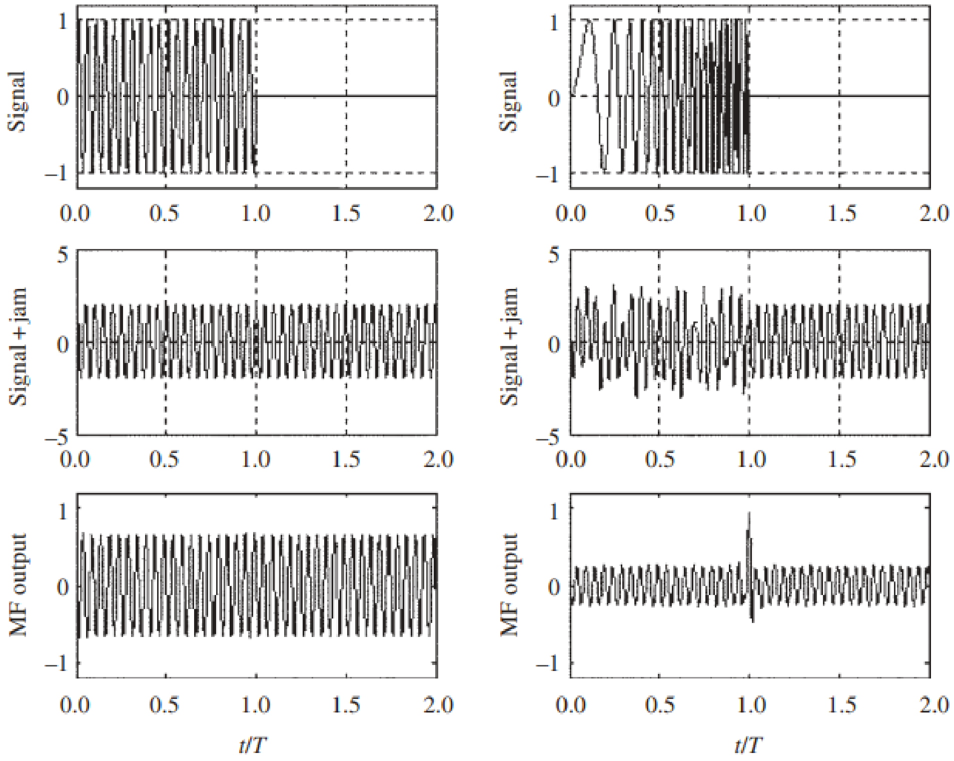


Figure 6 - Comparison between a rectangular and a spread spectrum signal in the presence of a barrage jammer [18]

The second one (b), the partial band jammer, is a jammer that only transmits bandlimited AWGN, covering only a part of the signal's bandwidth. The advantage of these over the barrage jammers is that it can channel more power over one or more desired sections of the signal's bandwidth, especially if dealing with frequency hopping spread spectrum signals [19].

Tone jamming, the third (c) and fourth (d) on Figure 2, happens when one or more tones are introduced in the signal's spectrum. Their jamming efficiency depends on the number of tones and on their location (considering the same Power Spectral Density (PSD) of the previous). Typically, on an Unmanned Aerial System (UAS), a tactical jammer is limited to about 100 watts, making the number of tones to be limited to around 10 [15]. As stated before, tone jamming may be divided into single tone jamming and multitone jamming. Single tone jamming also known as a spot jamming, according to [20], is the jamming of a specific or frequency. It's basically a Continuous Wave (CW) signal transmitted on a single frequency. By deploying single tone jammers against

Direct-sequence spread spectrum (DSSS) signals, especially if close to the receiver, the jamming signal causes problems in the gain control systems of the received signal [15].

Multitone jammers are jammers that emit more than one tone, capable of covering a larger bandwidth than a single tone jammer or jamming several channels on a multiple channel receiver [21].

The matched spectral jammer detects the signal's spectral density and uses a Gaussian random process to create a "matched spectral" jamming signal identical to the original signal [17].

Another way of dividing jammers by type is suggested in [22], for jammers in wireless networks, presented on Figure 7 and detailed on Tables 2, 3, 4 and 5:

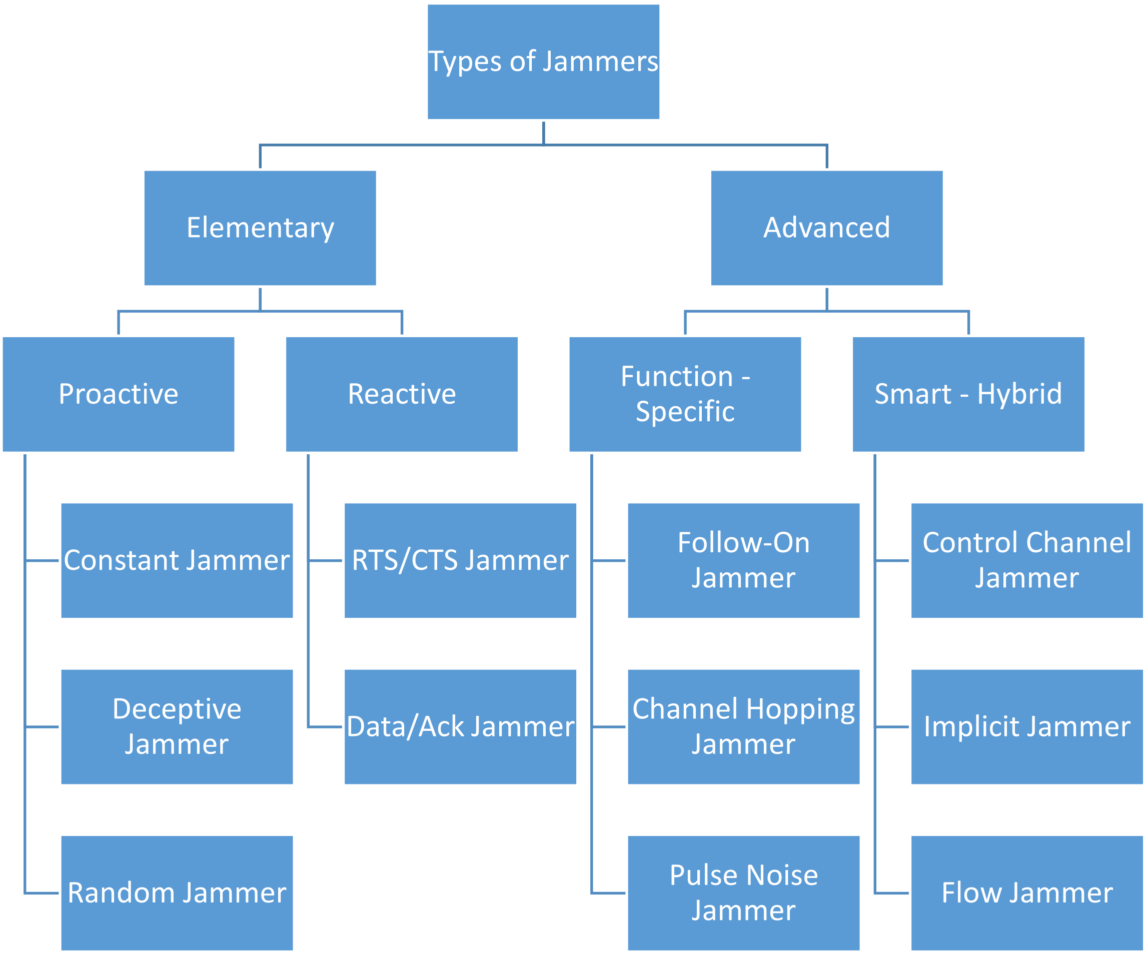


Figure 7 - Types of jammers by [12]

Table 2 - Proactive Jammers

Proactive Jammers	
Jammer that transmits whether there is or there is not data communication.	
Constant Jammer	Emits random bits continuously
Deceptive Jammer	Emits regular packets continuously
Random Jammer	Emits bits and/or packets intermittently

Table 3 - Reactive Jammers

Reactive Jammer	
Jammer that only transmits if data communication is detected	
RTS/CTS Jammer	Jams when requests-to-send (RTS) signal is emitted. This prevents de receiver to send the clear-to-send (CTS) signal because the RTS signal is distorted
Data/ACK Jammer	Corrupts the data or acknowledgment packets

Table 4 - Function-specific Jammers

Function-specific Jammer	
Jammer implemented by having a pre-determined function. It can jam one or multiple channels	
Follow-on Jammer	Hops over all available channels, jamming each channel for a short period of time
Channel-hopping Jammer	Hops between different channels, with the possibility of jamming several channels at the same time
Pulsed-noise Jammer	It can switch channels and jam different bandwidths. Like the random jammer but with the possibility of attacking several channels

Table 5 - Smart-hybrid Jammers

Smart-hybrid Jammer Jammer with the capacity of conserving energy by emitting only the necessary power to disrupt the channel (smart). It can be either proactive or reactive (hybrid)	
Control Channel Jammer	Multi-channel jammer that targets the control channel used to coordinate the network activity
Implicit Jammer	Jammer that disables the functionality of the target and causes denial-of-service, by targeting the nodes of the access point
Flow-jamming Jammer	Array of jammers that jam packets in order to reduce traffic flow

2.4 Identification of the bands, systems and applicability on Naval Operations

In this sub-chapter, there are going to be presented the different bands and systems where the jammer works.

2.4.1 AIS

The Automatic Identification System (AIS) is a Very High Frequency (VHF) transponder system with an automatic identification capacity, developed by the International Maritime Organization, in the sequence of the revision of Chapter V of the Safety of Life at Sea (SOLAS) Convention [23]. It was designed as a navigation safety aid and as an environmental protection aid, through the exchange of information between ships and coast stations [24].

Because this system is mandatory for every ship with 300 or more gross tonnage and all passenger vessels and highly recommended for all ships, almost every vessel uses this system [25]. Also, according to the MarineTraffic website [26], AIS information is used to serve several purposes such as tug operations, communication between vessel crews and their family's members, coast guard and border patrol operations, environmental protection agencies, among others.

The AIS transmits a great variety of information about the ship such as: GPS data (position, course and speed), data from the gyrocompass, International Maritime Organization (IMO) number, call sign, length and beam, location of the antenna, ship type and draft, hazardous cargo, destination, Estimated Time of Arrival (ETA) and route plan. Its data that can be programmed into the unit because it never changes is known as static data like the IMO number, the data introduced by the user is known as voyage related data, like the route plan and the data that's being constantly updated (automatically) is known as dynamic data, like the ships' position.

The AIS works on VHF channel 87B and 88B at 161.975 MHz and 162.025 MHz respectively with a transmission bit rate of 9600 baud. It uses Self Organizing Time-Division Multiple Access in order to avoid collisions between the transmissions of the different stations, using the GPS time as a common time reference. AIS messages are standardized hence, it's relatively easy to create messages or to change their content directly on the raw code [27]. This makes them rather vulnerable to jamming and spoofing, and thereby, a good and easy target to perform electronic attacks. There are tactical situations where it could be useful to create "ghost contacts" in order to baffle the enemy's perception of the surrounding environment. On the other hand, it might also be useful to deny them all AIS information by jamming the system.

2.4.2 ISM Band

The Industrial, Scientific and Medical (ISM) band are parts of the EM spectrum allocated for Radio Frequency (RF) use of industrial, scientific and medical equipment. According to the International Telecommunications Union (ITU) [28], ISM band application is the "Operation of equipment or appliances designed to generate and use locally radio frequency energy for industrial, scientific, medical, domestic or similar purposes, excluding applications in the field of telecommunications." In the definition by ITU, telecommunication applications are not considered because of the RF interference that ISM equipment may cause but, since this band is open and unlicensed, there are also many short-range telecommunications devices using this band.

According to the European Communications Committee [29], the ISM band frequencies in Europe are presented on Table 6:

Table 6 - ISM Band Frequencies

ISM BAND FREQUENCIES	
Start Frequency	End Frequency
11.3 kHz	14 kHz
6765 kHz	6795 kHz
13553 kHz	13567 kHz
26.957 MHz	27.283 MHz
40.66 MHz	40.7 MHz
430.05 MHz	434.79 MHz
2400 MHz	2500 MHz
5725 MHz	5875 MHz
24 GHz	24.25 GHz
61 GHz	61.5 GHz

Many systems that we use on a daily basis work on the ISM band. Some examples of these systems are the IEEE 802.11/Wi-Fi, Bluetooth or car remote controls [30].

2.4.3 GNSS

The satellite navigation is a very important component of modern navigation systems, specially the systems contained in the Global Navigation Satellite System (GNSS). Among the existing GNSSs, three were explored: the GPS, the Globalnaya navigatsionnaya sputnikovaya Sistema (GLONASS) and the Galileo [31].

2.4.3.1 GPS

The GPS is, according to the United States (US) Government [32], a U.S.-owned utility that provides users with positioning, navigation, and timing (PNT) services. The system uses satellites in order to provide the PNT information. The GPS has many applications worldwide from navigation to clock sensitive systems, which makes it essential for some systems to work properly.

The GPS satellites emit two signals: a civilian which apart of some restrictions, its use is authorized for everyone and a military which is encrypted by the U.S. Department of Defence. The GPS civilian signal works in two frequencies: the L1 and the L2C. The L1 and the L2C are modulated with a carrier wave that works in the frequency 1575.2 MHz and 1227.6 MHz and, just like the AIS signal, its packet information is public, making it vulnerable to electronic attacks [33]. The tactical and strategical advantages of controlling the GPS signal are significant. For example, by denying the enemy the capacity of receiving a GPS signal, several systems are affected, reducing their combat and defence capacity.

2.4.3.2 GLONASS

The GLONASS is a Russian satellite navigation system like the GPS. It was developed by the Russian Federal Space Agency and also has both a military and a civilian signal. The civilian signals' frequency is 1602+n*0.5625 MHz and 1246+n*0.4375 MHz, being n the channel number [34].

2.4.3.3 GALILEO

Just like the US and Russia had the necessity of having an independent satellite navigation system, the European Union also developed its own. Galileo, named after Galileo Galilei, is the European satellite navigation system developed by the European Global Navigation Satellite System Agency [35].

3 Software, Hardware and Interface Development

In this chapter, there are presented all the necessary software development and the hardware solutions founded for the creation of the EW package.

3.1 Definition of the Hardware

In this sub-chapter, there will be presented the architecture for the EW package.

3.1.1 SDR

According to different organizations, there are many definitions for Software Defined Radio (SDR) [36]. For example, the ITU defines it as “A radio in which the RF operating parameters of frequency range, modulation type, and/or output power can be set or altered by software, or the technique by which this is achieved”[37] and the U.S. Federal Communications Commission defines it as “A radio that includes a transmitter in which the operating parameters of frequency range, modulation type or maximum output power (either radiated or conducted), or the circumstances under which the transmitter operates in accordance with Commission rules, can be altered by making a change in software without making any changes to hardware components that affect the radio frequency emissions”[38]. Although there are different definitions for SDR, they all state that SDR is a radio platform where we can alter several parameters in software without changing the hardware.

Basically, an SDR is a radio and software technique used in radio communications, where most or all of the signal processing, such as modulation, sampling and waveform synthesis, is done using software routines in general purpose processors (GPP) rather than application-specific hardware (like in hardware-based radios). Digital signal processing (DSP) made in software has several advantages over DSP made in hardware, including higher flexibility and adaptability (routines can be changed during execution) and a reduction of cost inherent to the hardware since general purpose hardware is used. Other two important advantages of using SDRs is the existence of a big online community, offering lots of support (specially for the open source SDRs) and making it accessible and easy to work with, even for people with basic knowledge of signal processing, and the available open source DSP frameworks tailored for SDR applications.

3.1.2 HackRF One

The HackRF one is an open-source SDR half-duplex transceiver designed by Michael Ossmann [39], with operating frequencies from 1 MHz to 6 GHz. It's able to process 20 million samples per second (being its maximum bandwidth 20 MHz), it connects to a computer through USB and it's compatible with GNU Radio [40]. Its maximum transmission power is [39]:

- 10 MHz to 2150 MHz: 5 dBm to 15 dBm, generally increasing as frequency decreases
- 2150 MHz to 2750 MHz: 13 dBm to 15 dBm
- 2750 MHz to 4000 MHz: 0 dBm to 5 dBm, increasing as frequency decreases
- 4000 MHz to 6000 MHz: -10 dBm to 0 dBm, generally increasing as frequency decreases

The HackRF One was chosen once it was the one with a better relationship between price, operating frequencies, bandwidth and samples rate when compared to its peers. A brief comparison between the most common RX/TX SDRs is made on Table 7:

Table 7 - Comparison between SDRs

Equipment	Operating Frequency	Bandwidth	Price (Approx.)	Sample Rate
HackRF One	1 MHz to 6 GHz	20 MHz	150€	20 MHz
LimeSDR	100 KHz to 3.8 GHz	61.44 MHz	250€	61.44 MHz
USRP B200	70 MHz to 6 GHz	56 MHz	600€	56 MHz
ADALM-PLUTO	325 MHz to 3.8 MHz	20 MHz	150€	61.44 MHz

3.1.3 GNU Radio

According to the definition on [41], "GNU Radio is a free & open-source software development toolkit that provides signal processing blocks to implement software radios. It can be used with readily available low-cost external RF hardware to create software-defined radios, or without hardware in a simulation-like environment. It is widely used in research, industry, academia, government, and hobbyist environments to support both wireless communications research and real-world radio systems."

GNU Radio provides a useful tool called GNU Radio Companion. The Companion is a tool that allows the user to build flow graphs through a series of pre-made blocks, providing a way to create its programs in an intuitive way, with a graphical interface, for rapid prototyping. Because the Companion's blocks generate a python script, it's possible to program the blocks with python code lines, allowing the blocks to have a greater interaction between them, making it a powerful signal processing tool.

3.1.4 Raspberry Pi

To run the previous programs, the HackRF must be connected to a computer containing the programs and the necessary dependencies. In order to implement them on an autonomous vehicle, especially on an Unmanned Aerial Vehicle (UAV), the computer must be small and light without losing the processing capacity to run the programs.

To accomplish this, the found solution was to use a Raspberry Pi. The Raspberry Pi is a low-cost single-board computer built by the Raspberry Pi Foundation [42] which, according to the latter, is intended for code learning and to build electronics projects. The used model is the Raspberry Pi 3B+ with a 1 GB RAM, 4 USB ports and 1.4 GHz 64-bit CPU. Its storage is guaranteed by a 32 GB SD card with the Raspbian Operating System installed [43].

The Raspberry was chosen because it was the one with a better relationship between price, CPU, RAM and size of the online community, when compared to its peers. A brief comparison between the most common single board computers is made on Table 8:

Table 8 - Comparison between single board computers

Equipment	Price (Approx.)	CPU	RAM	Online Community
Raspberry Pi 3B+	35€	ARM Cortex-A53	1 GB	Big
Banana Pi	35€	ARM Cortex-A7	1 GB	Medium
ODROID-XU4	70€	ARM Cortex-A15	2GB	Medium

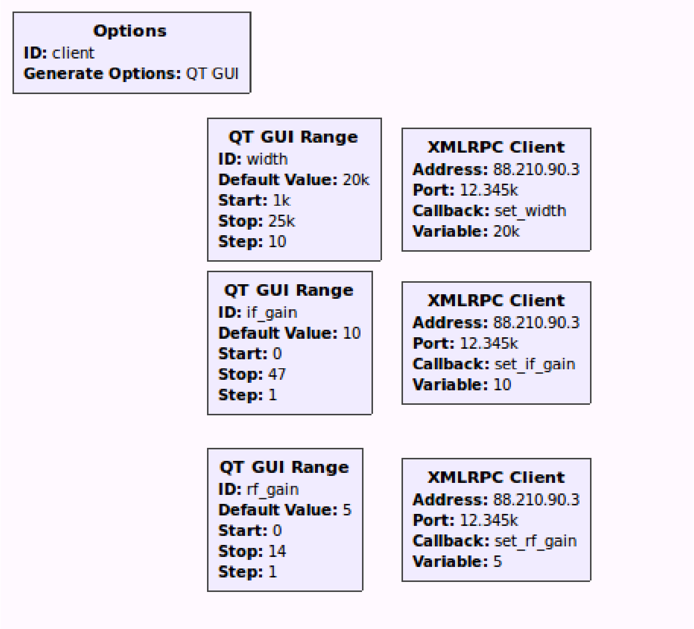
3.2 Software Development

To create the jamming and spoofing capacities that compose the EW package, GNU Radio programs were developed. All programs were divided in two parts: a client and a server. The server is the part that runs on the autonomous vehicle and the client is the part that is on the Ground Control Station (GCS).

3.2.1 AIS Jammer:

The AIS Jammer is a program composed by a set of blocks with the capacity of jamming an AIS signal reducing the situational awareness for the jammed unit. The blocks provide the user the options of choosing the bandwidth of the jamming signal adapted to the AIS and the IF and RF gains. The AIS Jammer was built in GNU Radio with the blocks on Figures 8 and 9:

3.2.1.1 Client:



Options
ID: client
Generate Options: QT GUI

QT GUI Range
ID: width
Default Value: 20k
Start: 1k
Stop: 25k
Step: 10

XMLRPC Client
Address: 88.210.90.3
Port: 12.345k
Callback: set_width
Variable: 20k

QT GUI Range
ID: if_gain
Default Value: 10
Start: 0
Stop: 47
Step: 1

XMLRPC Client
Address: 88.210.90.3
Port: 12.345k
Callback: set_if_gain
Variable: 10

QT GUI Range
ID: rf_gain
Default Value: 5
Start: 0
Stop: 14
Step: 1

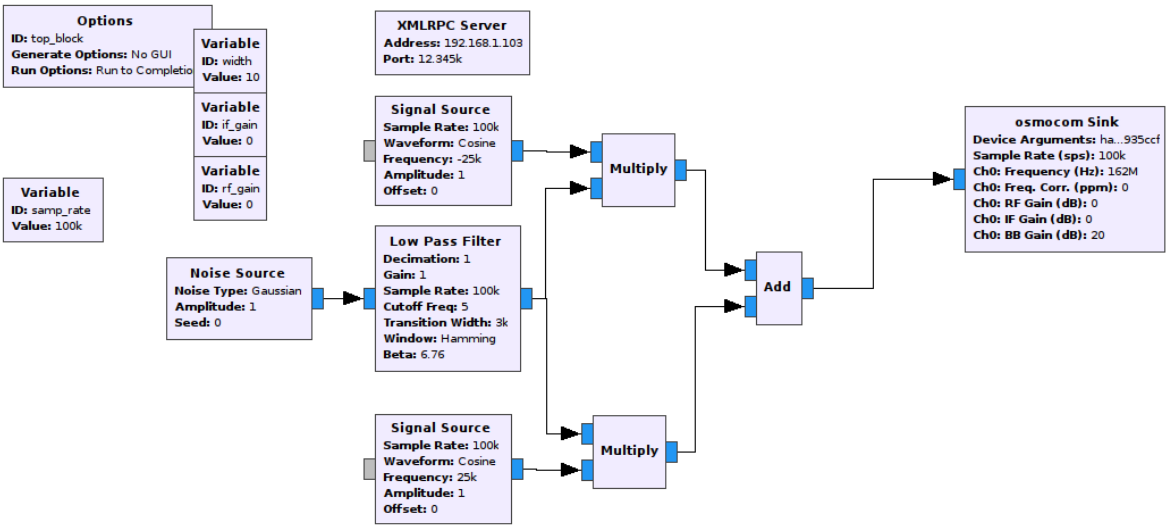
XMLRPC Client
Address: 88.210.90.3
Port: 12.345k
Callback: set_rf_gain
Variable: 5

Figure 8 - AIS Jammer Client Block Diagram

The used blocks were:

- QT GUI Range ID: “width” – This block is a slider which gives the user the option of controlling the width of the jamming signal by changing the cut-off frequency of the low pass filter. It has a minimum value of 1 kHz, a maximum value of 25 kHz and a default value of 20 kHz.
- QT GUI Range ID: “if_gain” and “rf_gain” – These blocks are sliders that are used to control the intermediate frequency (IF) gain and the RF gain. The IF gain slider has a minimum of 0 dB, a maximum of 47 dB (HackRF limit) and a default of 10dB and the RF slider has a minimum of 0 dB, a maximum of 14 dB (HackRF limit) and a default of 5 dB.
- XMLRPC Client – Block that makes the connection between the Server and Client using the XML-RPC protocol. Each variable has a dedicated XMLRPC block which has as input the Internet Protocol (IP) address and port of the machine where the Server is.

3.2.1.2 Server:



Options
ID: top_block
Generate Options: No GUI
Run Options: Run to Completio

Variable
ID: width
Value: 10

Variable
ID: if_gain
Value: 0

Variable
ID: rf_gain
Value: 0

Variable
ID: samp_rate
Value: 100k

XMLRPC Server
Address: 192.168.1.103
Port: 12.345k

Signal Source
Sample Rate: 100k
Waveform: Cosine
Frequency: -25k
Amplitude: 1
Offset: 0

Noise Source
Noise Type: Gaussian
Amplitude: 1
Seed: 0

Low Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 100k
Cutoff Freq: 5
Transition Width: 3k
Window: Hamming
Beta: 6.76

Signal Source
Sample Rate: 100k
Waveform: Cosine
Frequency: 25k
Amplitude: 1
Offset: 0

osmocom Sink
Device Arguments: ha...935ccf
Sample Rate (sps): 100k
Ch0: Frequency (Hz): 162M
Ch0: Freq. Corr. (ppm): 0
Ch0: RF Gain (dB): 0
Ch0: IF Gain (dB): 0
Ch0: BB Gain (dB): 20

Figure 9 - AIS Jammer Server Block Diagram

The used blocks were:

- Variable ID: “samp_rate” – Variable containing the sample rate used by the other blocks. It has a value of 100 kHz which is enough to cover both AIS Channels.
- Variable ID: “if_gain” – Variable that receives the desired value to the IF gain.
- Variable ID: “rf_gain” – Variable that receives the desired value to the RF gain.
- Variable ID: “width” – Variable that receives the desired value to the bandwidth of the jamming signal.
- XMLRPC Server - Block that makes the connection between the Server and Client using the XML-RPC protocol. The XMLRPC block is programmed with the Internet Protocol (IP) address and port of the Server.
- Noise Source – Gaussian noise source which will produce the jamming signal.
- Low Pass Filter – Low pass filter with a cut-off frequency equal to “width/2” and a transition width of 3 kHz. This block is used to filter the noise to a bandwidth equal to the bandwidth of each AIS channel. The graph of the low pass filter is shown on Figure 10:
- Signal Source – Two cosine signal sources, with a frequency of -25 kHz and 25 kHz. These frequencies are used because the two AIS channels are located at 162 MHz ± 25kHz. When they are multiplied by the signal generated by the low pass

filter, they produce two waves with a bandwidth equal to “width/2” centred at ± 25 kHz. The equation that represents the generated signal is the following, with A being the amplitude and f the difference between the channel frequencies and 162MHz:

$$y(t) = A\cos(2\pi \pm f) \tag{1}$$

- Add – Block used to add the ± 25 kHz signals.
- Osmocom Sink – Block that sends the information to the HackRF and shifts the signals to the desired frequency. In this case, it’s set to 162 MHz, the AIS frequency. It also deals with the IF and RF gains provided by the sliders.

Figure 10 shows the information sent by the “Add” block to the HackRF with the “width” variable with the value 10.

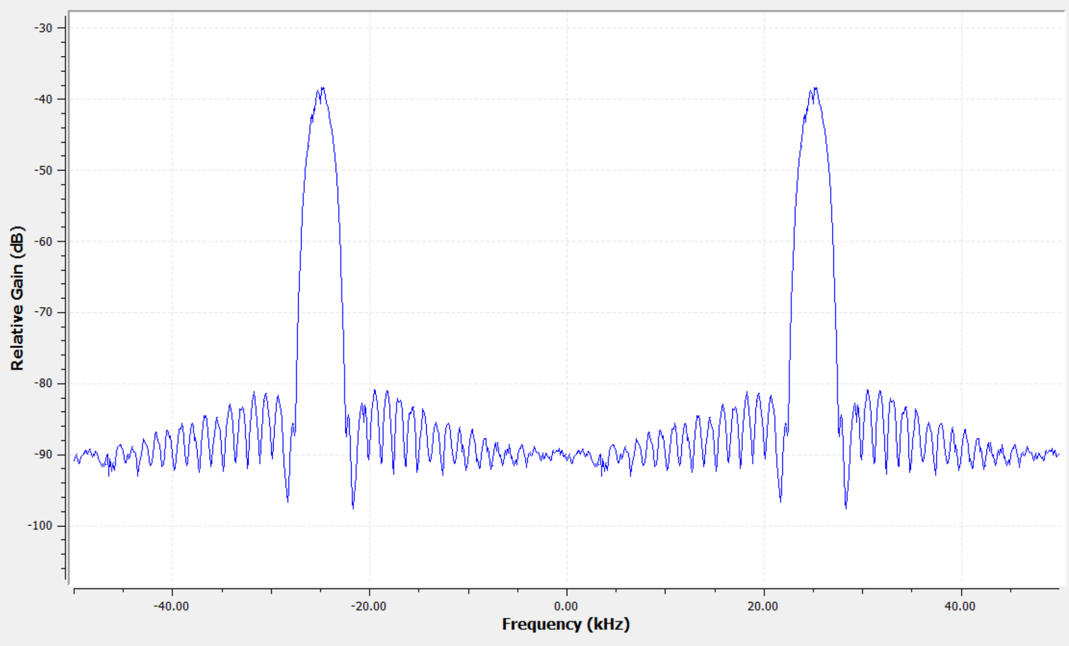
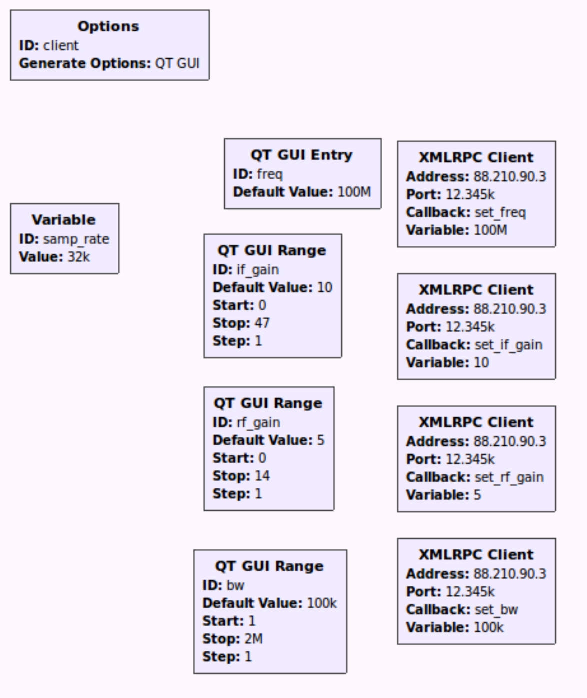


Figure 10 - Information sent to the HackRF by the “Add” Block

3.2.2 Free Jammer

The Free Jammer is a program with the objective of generating a jamming signal in an arbitrary frequency. The user can choose the centre frequency of the jammer, between 30 MHz and 6 GHz (operating frequencies of the HackRF), the bandwidth of the jamming signal and control the IF and RF gains. The Free Jammer was built in GNU Radio with the blocks on Figures 11 and 12:

3.2.2.1 Client:



Options
ID: client
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 32k

QT GUI Entry
ID: freq
Default Value: 100M

XMLRPC Client
Address: 88.210.90.3
Port: 12.345k
Callback: set_freq
Variable: 100M

QT GUI Range
ID: if_gain
Default Value: 10
Start: 0
Stop: 47
Step: 1

XMLRPC Client
Address: 88.210.90.3
Port: 12.345k
Callback: set_if_gain
Variable: 10

QT GUI Range
ID: rf_gain
Default Value: 5
Start: 0
Stop: 14
Step: 1

XMLRPC Client
Address: 88.210.90.3
Port: 12.345k
Callback: set_rf_gain
Variable: 5

QT GUI Range
ID: bw
Default Value: 100k
Start: 1
Stop: 2M
Step: 1

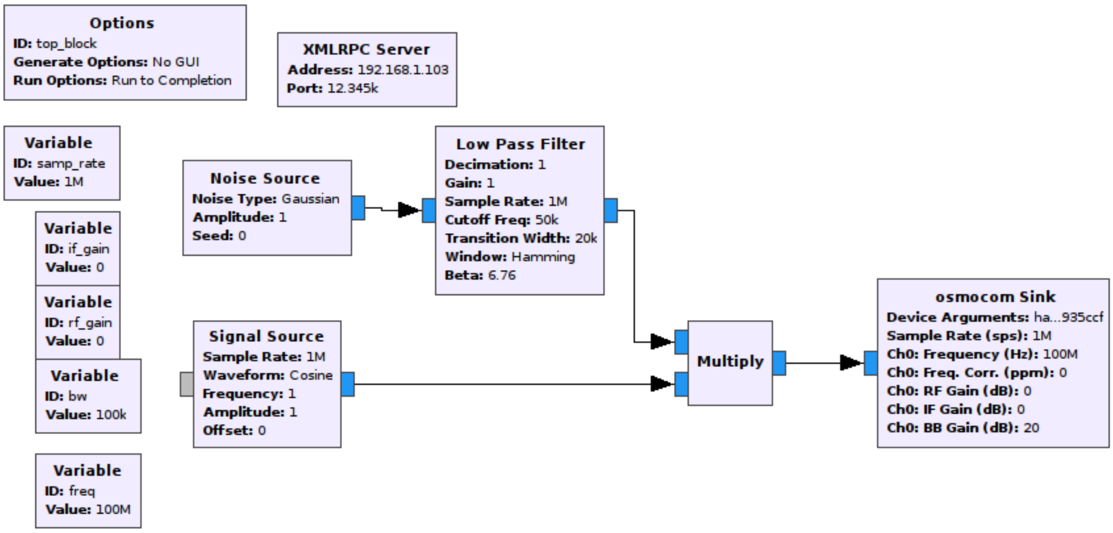
XMLRPC Client
Address: 88.210.90.3
Port: 12.345k
Callback: set_bw
Variable: 100k

Figure 11 - Free Jammer Client Block Diagram

The used blocks were:

- Variable ID: “samp_rate” – Used for tests.
- QT GUI Entry ID: “freq” – Block that generates a text box for the user to introduce the desired centre frequency.
- QT GUI Range ID: “bw” – Slider to control the bandwidth of the emitted signal. It has a minimum of 1 Hz, a maximum of 2 MHz (might be altered according to the purpose of the jammer) and a default of 100 kHz.
- XMLRPC Client; QT GUI Range ID: “if_gain” and “rf_gain” – Same function as in 3.2.1.1

3.2.2.2 Server:



Options
ID: top_block
Generate Options: No GUI
Run Options: Run to Completion

XMLRPC Server
Address: 192.168.1.103
Port: 12.345k

Variable
ID: samp_rate
Value: 1M

Variable
ID: if_gain
Value: 0

Variable
ID: rf_gain
Value: 0

Variable
ID: bw
Value: 100k

Variable
ID: freq
Value: 100M

Noise Source
Noise Type: Gaussian
Amplitude: 1
Seed: 0

Low Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 1M
Cutoff Freq: 50k
Transition Width: 20k
Window: Hamming
Beta: 6.76

Signal Source
Sample Rate: 1M
Waveform: Cosine
Frequency: 1
Amplitude: 1
Offset: 0

Multiply

osmocom Sink
Device Arguments: ha...935ccf
Sample Rate (sps): 1M
Ch0: Frequency (Hz): 100M
Ch0: Freq. Corr. (ppm): 0
Ch0: RF Gain (dB): 0
Ch0: IF Gain (dB): 0
Ch0: BB Gain (dB): 20

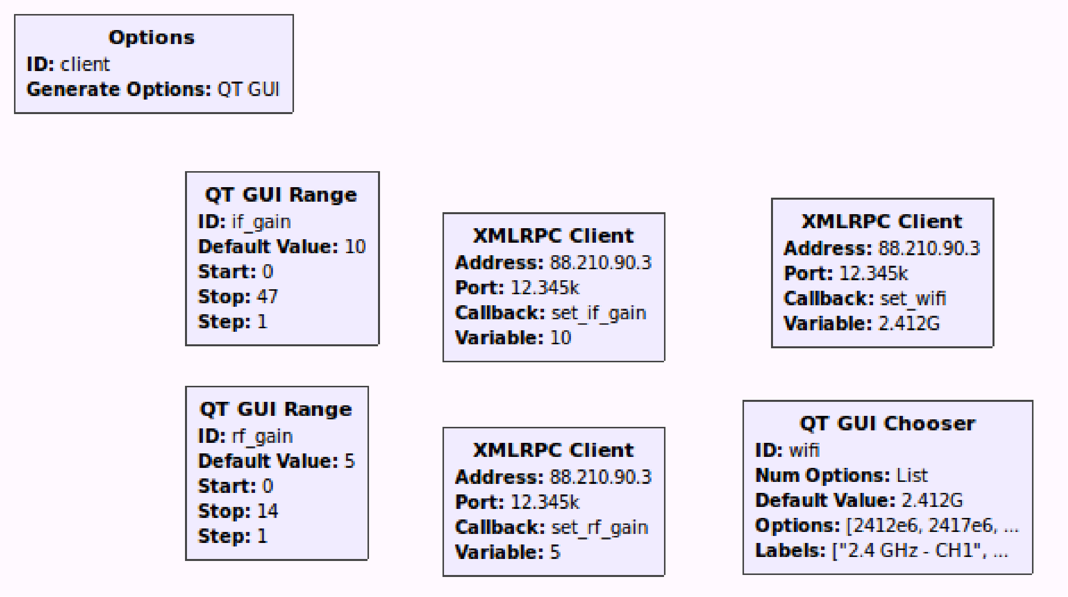
Figure 12 - Free Jammer Server Block Diagram

- Variables – Variables that receive the values from the client through the XMLRPC blocks.
- Multiply – Block used to multiply the signal from the "Signal Source" and the "Low Pass Filter".
- Low Pass Filter – Same function as in 3.2.1.2 with a cut-off frequency of "bandwidth/2".
- XMLRPC Server; Noise Source; Signal Source – Same function as in 3.2.1.2
- Osmocom sink – Same function as in 3.2.1.2 with a variable centre frequency defined by the "bw" variable.

3.2.3 Wi-Fi Jammer

The Wi-Fi Jammer is a program with the capacity of jamming signals that operate in the same frequency as the Wi-Fi. The user can choose which channel does he wish to jam in the 2.4GHz band and in the 5GHz band. The Wi-Fi Jammer was built with the blocks on Figures 13 and 14:

3.2.3.1 Client:



Options
ID: client
Generate Options: QT GUI

QT GUI Range
ID: if_gain
Default Value: 10
Start: 0
Stop: 47
Step: 1

QT GUI Range
ID: rf_gain
Default Value: 5
Start: 0
Stop: 14
Step: 1

XMLRPC Client
Address: 88.210.90.3
Port: 12.345k
Callback: set_if_gain
Variable: 10

XMLRPC Client
Address: 88.210.90.3
Port: 12.345k
Callback: set_rf_gain
Variable: 5

XMLRPC Client
Address: 88.210.90.3
Port: 12.345k
Callback: set_wifi
Variable: 2.412G

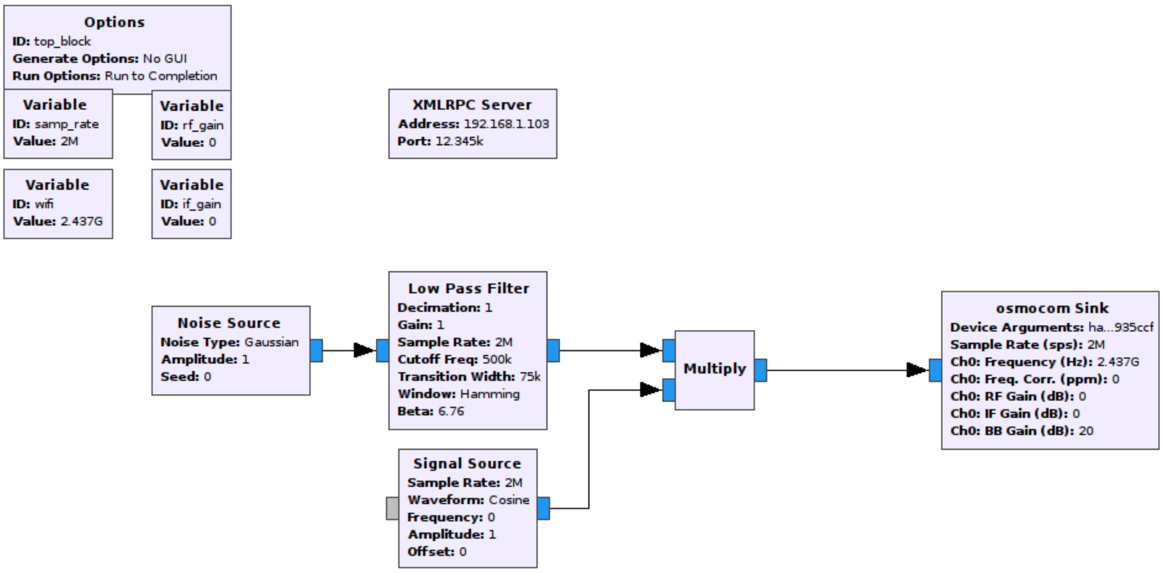
QT GUI Chooser
ID: wifi
Num Options: List
Default Value: 2.412G
Options: [2412e6, 2417e6, ...
Labels: ["2.4 GHz - CH1", ...

Figure 13 - Wi-Fi Jammer Client Block Diagram

The used blocks were:

- QT GUI Chooser ID: “wifi” – Block that allows the user to choose which Wi-Fi channel does he want to jam.
- XMLRPC Client; QT GUI Range ID: “if_gain” and “rf_gain” – Same function as in 3.1.1.1

3.2.3.2 Server:



Options
ID: top_block
Generate Options: No GUI
Run Options: Run to Completion

Variable
ID: samp_rate
Value: 2M

Variable
ID: rf_gain
Value: 0

Variable
ID: wifi
Value: 2.437G

Variable
ID: if_gain
Value: 0

XMLRPC Server
Address: 192.168.1.103
Port: 12.345k

Noise Source
Noise Type: Gaussian
Amplitude: 1
Seed: 0

Low Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 2M
Cutoff Freq: 500k
Transition Width: 75k
Window: Hamming
Beta: 6.76

Signal Source
Sample Rate: 2M
Waveform: Cosine
Frequency: 0
Amplitude: 1
Offset: 0

Multiply

osmocom Sink
Device Arguments: ha...935ccf
Sample Rate (sps): 2M
Ch0: Frequency (Hz): 2.437G
Ch0: Freq. Corr. (ppm): 0
Ch0: RF Gain (dB): 0
Ch0: IF Gain (dB): 0
Ch0: BB Gain (dB): 20

Figure 14 - Wi-Fi Jammer Server Block Diagram

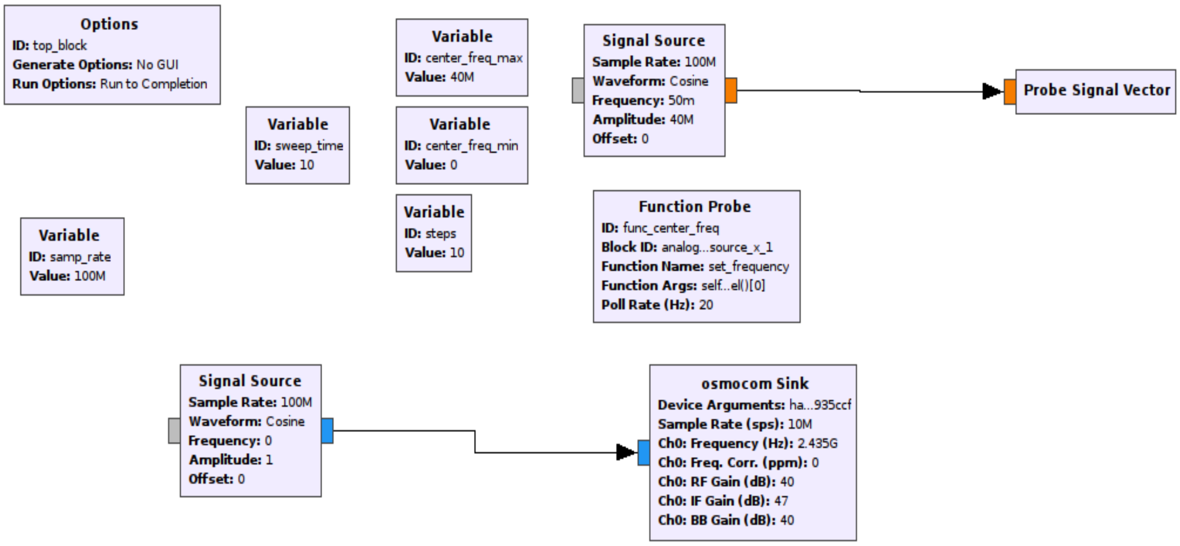
The used blocks were:

- Variables – Variables that receive the values from the client through the XMLRPC blocks.
- Multiply – Block used to multiply the signal from the "Signal Source" and the "Low Pass Filter".
- Low Pass Filter – Same function as in 3.2.1.2, with a cut-off frequency of 500kHz.
- XMLRPC Server; Noise Source; Signal Source – Same function as in 3.2.1.2.
- Osmocom sink – Same function as in 3.2.1.2, with a variable centre frequency defined by the "wifi" variable.

3.2.4 WI-FI Jammer All Band

The Wi-Fi Jammer All Band is a program designed to jam all the 2.4GHz Wi-Fi band. It only requires a Server because there are no changeable parameters, as shown on Figure 15.

3.2.4.1 Server



Options
ID: top_block
Generate Options: No GUI
Run Options: Run to Completion

Variable
ID: center_freq_max
Value: 40M

Signal Source
Sample Rate: 100M
Waveform: Cosine
Frequency: 50m
Amplitude: 40M
Offset: 0

Probe Signal Vector

Variable
ID: sweep_time
Value: 10

Variable
ID: center_freq_min
Value: 0

Variable
ID: samp_rate
Value: 100M

Variable
ID: steps
Value: 10

Function Probe
ID: func_center_freq
Block ID: analog...source_x_1
Function Name: set_frequency
Function Args: self...el()[0]
Poll Rate (Hz): 20

Signal Source
Sample Rate: 100M
Waveform: Cosine
Frequency: 0
Amplitude: 1
Offset: 0

osmocom Sink
Device Arguments: ha...935ccf
Sample Rate (sps): 10M
Ch0: Frequency (Hz): 2.435G
Ch0: Freq. Corr. (ppm): 0
Ch0: RF Gain (dB): 40
Ch0: IF Gain (dB): 47
Ch0: BB Gain (dB): 40

Figure 15 - Wi-Fi Jammer All Band Server Block Diagram

The Wi-Fi Jammer All Band Server is composed by the following blocks:

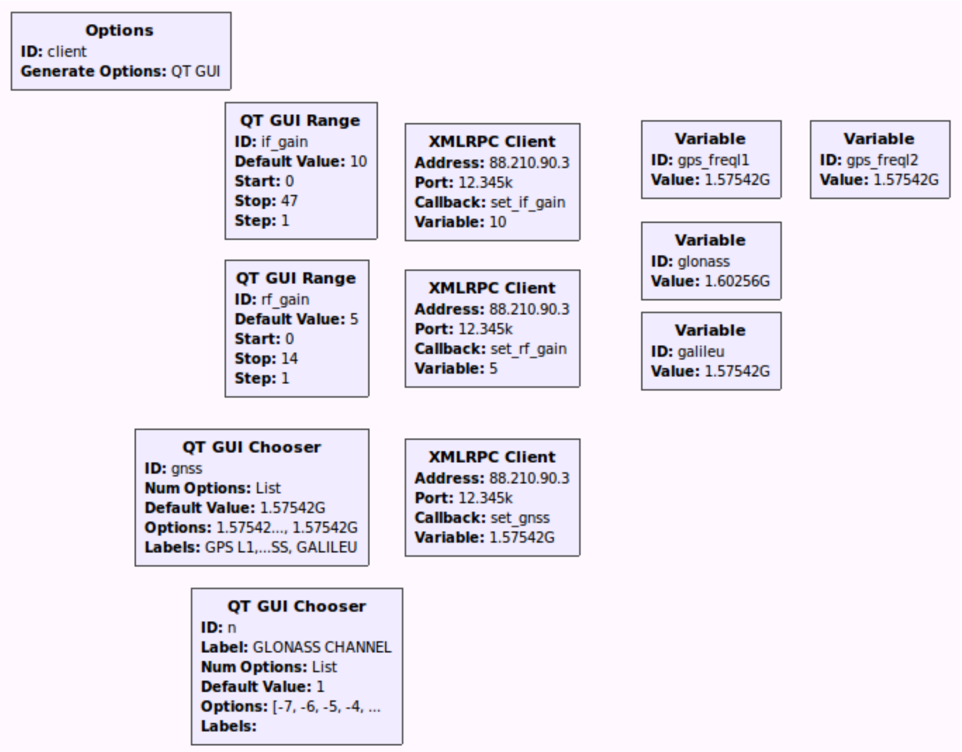
- Variable ID: "sweep_time" – Variables that defines a value for the sweep time on the "Function Probe" block.
- Variable ID: "steps" – Defines the number of steps taken on the sweep.

- Variables ID: “center_freq_max” and “center_freq_min” – Variables created to indicate the amplitude of the signal to be probed.
- Signal Source and Probe Signal Vector – The first block creates a cosine wave that is then probed by the Probe Signal Vector block.
- Function Probe – Receives the vector from the “Probe Signal Vector” block and creates a function with that data.
- Signal Source and Osmocom Sink – Receive the signal from the “Function Probe” block and transmits it to the HackRF.

3.2.5 GNSS Jammer

The GNSS Jammer is a program designed to jam three GNSSs: GPS, GLONASS and GALILEU. The GNSS Jammer has the block diagrams shown in Figures 16 and 17:

3.2.5.1 Client



Options
ID: client
Generate Options: QT GUI

QT GUI Range
ID: if_gain
Default Value: 10
Start: 0
Stop: 47
Step: 1

XMLRPC Client
Address: 88.210.90.3
Port: 12.345k
Callback: set_if_gain
Variable: 10

Variable
ID: gps_freql1
Value: 1.57542G

Variable
ID: gps_freql2
Value: 1.57542G

QT GUI Range
ID: rf_gain
Default Value: 5
Start: 0
Stop: 14
Step: 1

XMLRPC Client
Address: 88.210.90.3
Port: 12.345k
Callback: set_rf_gain
Variable: 5

Variable
ID: glonass
Value: 1.60256G

Variable
ID: galileu
Value: 1.57542G

QT GUI Chooser
ID: gnss
Num Options: List
Default Value: 1.57542G
Options: 1.57542..., 1.57542G
Labels: GPS L1,...SS, GALILEU

XMLRPC Client
Address: 88.210.90.3
Port: 12.345k
Callback: set_gnss
Variable: 1.57542G

QT GUI Chooser
ID: n
Label: GLONASS CHANNEL
Num Options: List
Default Value: 1
Options: [-7, -6, -5, -4, ...
Labels:

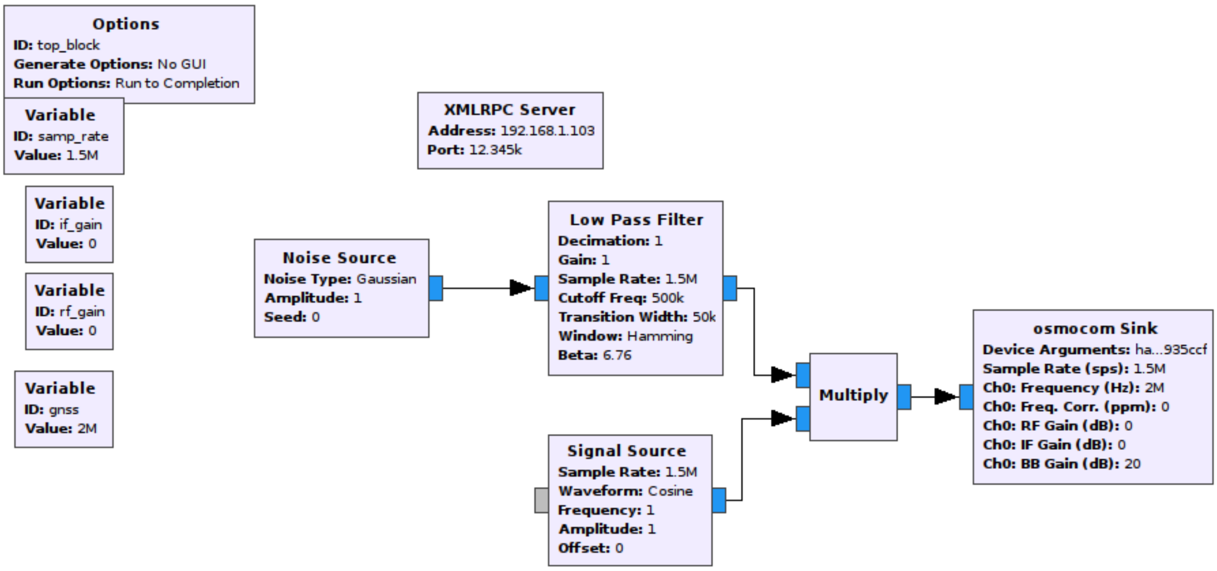
Figure 16 - GNSS Jammer Client Block Diagram

The used blocks were:

- Variable ID: “gps_freql1”, “gps_freql2”, “glonass” and “galileu” – Variables that define the centre frequency of the GPS, GLONASS and Galileo systems.

- QT GUI Chooser ID: “gnss” – Block that generates buttons that allow the user to choose which GNSS does he wish to jam.
- QT GUI Chooser ID: “n” – Block that creates a list of possible GLONASS channels to jam (-7 to 6).
- XMLRPC Client; QT GUI Range ID: “if_gain” and “rf_gain” – Same function as in 3.2.1.1

3.2.5.2 Server



Options
ID: top_block
Generate Options: No GUI
Run Options: Run to Completion

Variable
ID: samp_rate
Value: 1.5M

Variable
ID: if_gain
Value: 0

Variable
ID: rf_gain
Value: 0

Variable
ID: gnss
Value: 2M

XMLRPC Server
Address: 192.168.1.103
Port: 12.345k

Noise Source
Noise Type: Gaussian
Amplitude: 1
Seed: 0

Low Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 1.5M
Cutoff Freq: 500k
Transition Width: 50k
Window: Hamming
Beta: 6.76

Signal Source
Sample Rate: 1.5M
Waveform: Cosine
Frequency: 1
Amplitude: 1
Offset: 0

Multiply

osmocom Sink
Device Arguments: ha...935ccf
Sample Rate (sps): 1.5M
Ch0: Frequency (Hz): 2M
Ch0: Freq. Corr. (ppm): 0
Ch0: RF Gain (dB): 0
Ch0: IF Gain (dB): 0
Ch0: BB Gain (dB): 20

Figure 17 - GNSS Jammer Server Block Diagram

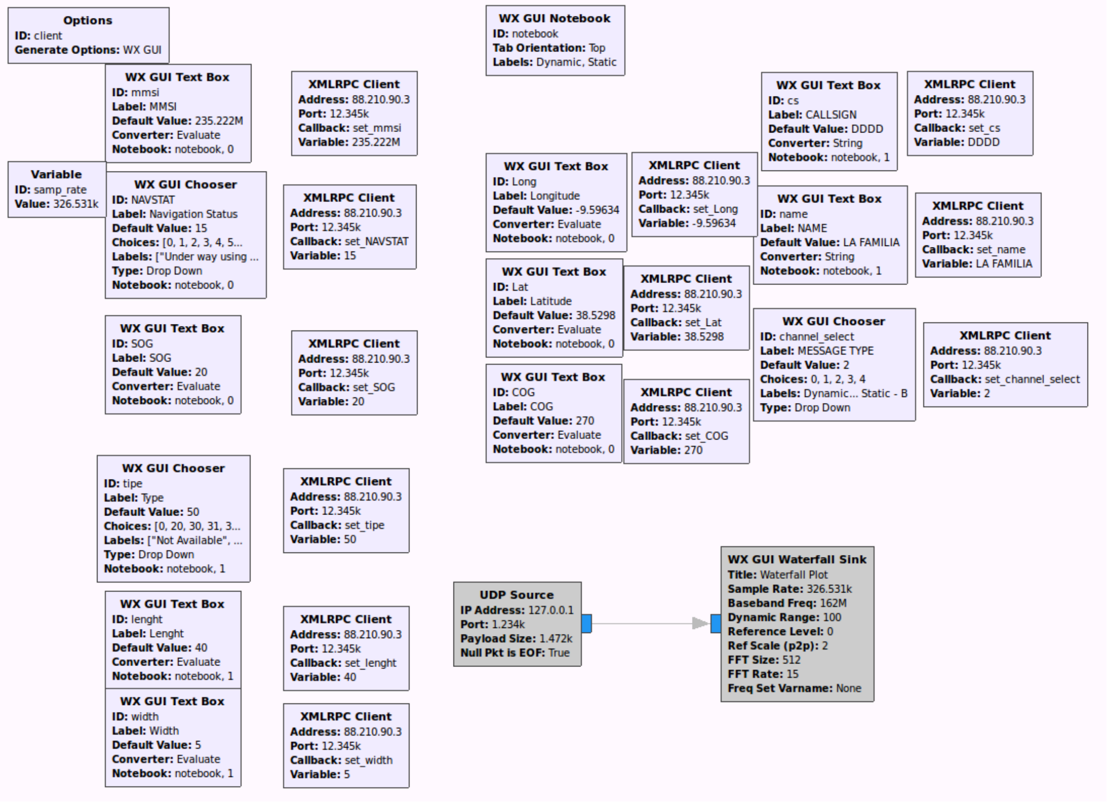
The used blocks were:

- Variables – Variables that receive the values from the client through the XMLRPC blocks.
- Multiply – Block used to multiply the signal from the “Signal Source” and the “Low Pass Filter”.
- XMLRPC Server; Noise Source; Signal Source; Low Pass Filter – Same function as in 3.2.1.2.
- Osmocom sink – Same function as in 3.2.1.2 with a variable centre frequency defined by the “gnss” variable.

3.2.6 AIS Spoofer

The AIS Spoofer is a program that has the capacity of creating AIS contacts. The AIS sentences on the server were built with the codes provided in [44] for AIVDM/AIVDO protocol decoding and by the US Coast Guard on [45] and the AIS Frame Builder Block used was created by Marco Balduzzi [46]. The AIS Spoofer has the following block diagrams (Figures 18 and 19):

3.2.6.1 Client



Options
ID: client
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 326.531k

WX GUI Text Box
ID: mmsi
Label: MMSI
Default Value: 235.222M
Converter: Evaluate
Notebook: notebook, 0

XMLRPC Client
Address: 88.210.90.3
Port: 12.345k
Callback: set_mmsi
Variable: 235.222M

WX GUI Chooser
ID: NAVSTAT
Label: Navigation Status
Default Value: 15
Choices: [0, 1, 2, 3, 4, 5...
Labels: ["Under way using ...
Type: Drop Down
Notebook: notebook, 0

XMLRPC Client
Address: 88.210.90.3
Port: 12.345k
Callback: set_NAVSTAT
Variable: 15

WX GUI Text Box
ID: SOG
Label: SOG
Default Value: 20
Converter: Evaluate
Notebook: notebook, 0

XMLRPC Client
Address: 88.210.90.3
Port: 12.345k
Callback: set_SOG
Variable: 20

WX GUI Chooser
ID: tipe
Label: Type
Default Value: 50
Choices: [0, 20, 30, 31, 3...
Labels: ["Not Available", ...
Type: Drop Down
Notebook: notebook, 1

XMLRPC Client
Address: 88.210.90.3
Port: 12.345k
Callback: set_tipe
Variable: 50

WX GUI Text Box
ID: lenght
Label: Lenght
Default Value: 40
Converter: Evaluate
Notebook: notebook, 1

XMLRPC Client
Address: 88.210.90.3
Port: 12.345k
Callback: set_lenght
Variable: 40

WX GUI Text Box
ID: width
Label: Width
Default Value: 5
Converter: Evaluate
Notebook: notebook, 1

XMLRPC Client
Address: 88.210.90.3
Port: 12.345k
Callback: set_width
Variable: 5

WX GUI Notebook
ID: notebook
Tab Orientation: Top
Labels: Dynamic, Static

WX GUI Text Box
ID: Long
Label: Longitude
Default Value: -9.59634
Converter: Evaluate
Notebook: notebook, 0

XMLRPC Client
Address: 88.210.90.3
Port: 12.345k
Callback: set_Long
Variable: -9.59634

WX GUI Text Box
ID: Lat
Label: Latitude
Default Value: 38.5298
Converter: Evaluate
Notebook: notebook, 0

XMLRPC Client
Address: 88.210.90.3
Port: 12.345k
Callback: set_Lat
Variable: 38.5298

WX GUI Text Box
ID: COG
Label: COG
Default Value: 270
Converter: Evaluate
Notebook: notebook, 0

XMLRPC Client
Address: 88.210.90.3
Port: 12.345k
Callback: set_COG
Variable: 270

WX GUI Text Box
ID: cs
Label: CALLSIGN
Default Value: DDDD
Converter: String
Notebook: notebook, 1

XMLRPC Client
Address: 88.210.90.3
Port: 12.345k
Callback: set_cs
Variable: DDDD

WX GUI Text Box
ID: name
Label: NAME
Default Value: LA FAMILIA
Converter: String
Notebook: notebook, 1

XMLRPC Client
Address: 88.210.90.3
Port: 12.345k
Callback: set_name
Variable: LA FAMILIA

WX GUI Chooser
ID: channel_select
Label: MESSAGE TYPE
Default Value: 2
Choices: 0, 1, 2, 3, 4
Labels: Dynamic... Static - B
Type: Drop Down

XMLRPC Client
Address: 88.210.90.3
Port: 12.345k
Callback: set_channel_select
Variable: 2

UDP Source
IP Address: 127.0.0.1
Port: 1.234k
Payload Size: 1.472k
Null Pkt is EOF: True

WX GUI Waterfall Sink
Title: Waterfall Plot
Sample Rate: 326.531k
Baseband Freq: 162M
Dynamic Range: 100
Reference Level: 0
Ref Scale (p2p): 2
FFT Size: 512
FFT Rate: 15
Freq Set Varname: None

Figure 18 - AIS Spoofer Client Block Diagram

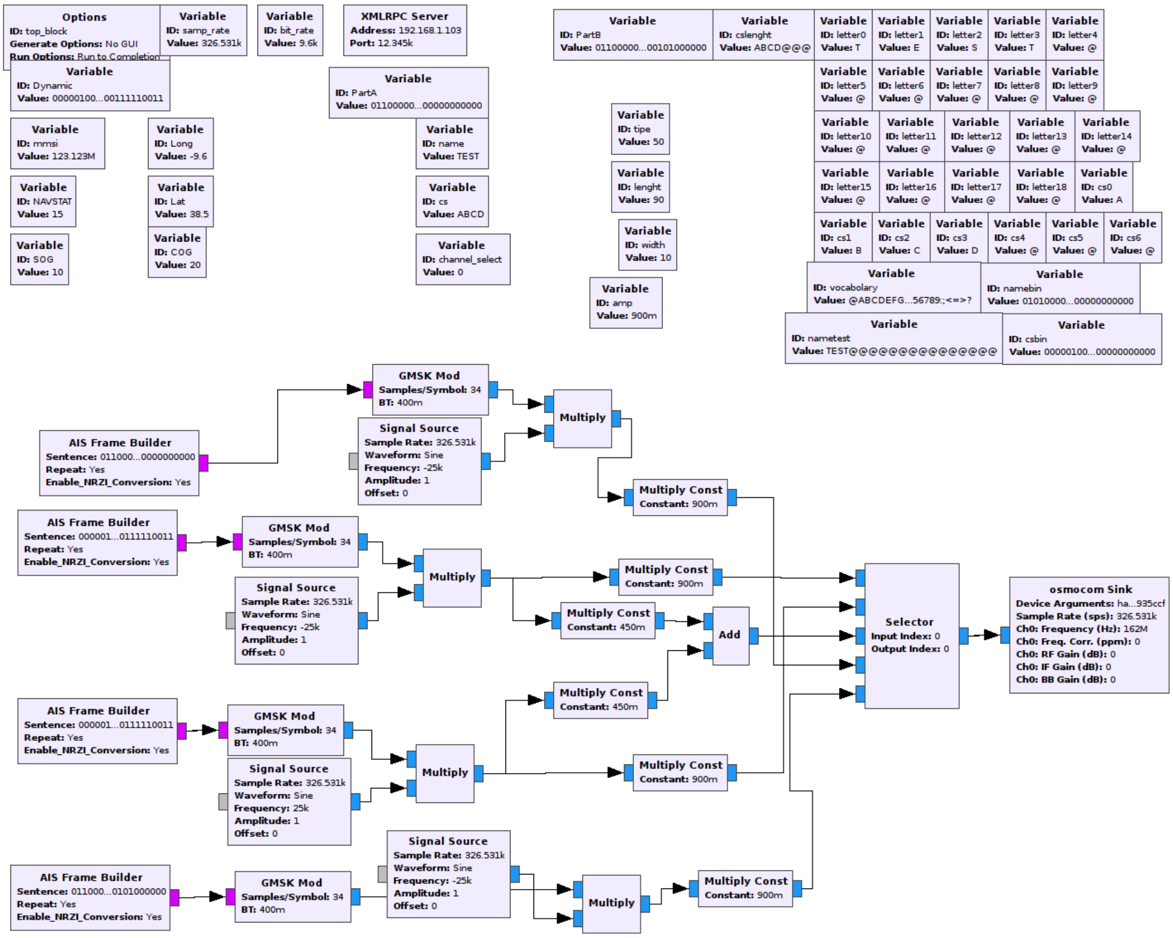
The used blocks were:

- WX GUI Text Box ID: “mmsi” – Block that receives the MMSI.
- WX GUI Chooser ID: “NAVSTAT” – Block that receives the navigational status and converts it into a number for the construction of the final message. The available navigational statuses are:
    - Under way using engine;

    - At anchor;
    - Not Under command;
    - Restricted manoeuvrability;
    - Moored;
    - Aground;
    - Engaged in fishing;
    - Under way sailing;
    - Not defined.
- WX GUI Text Box ID: “SOG” – Receives the speed over ground value in knots.
- WX GUI Chooser ID: “tipe” – Receives the type of ship and converts it into a number. The available types of ships are:
    - Not Available;
    - Wing in ground;
    - Fishing;
    - Towing;
    - Dredging or underwater ops;
    - Diving ops;
    - Military ops;
    - Sailing;
    - Pleasure Craft;
    - Pilot Vessel;
    - Search and Rescue vessel;
    - Tug;
    - Law Enforcement;
    - Medical Transport;
    - Non-combatant ship according to RR Resolution No. 18;
    - Passenger;
    - Cargo;
    - Tanker;
    - Other Type.
- WX GUI Text Box ID: “length” – Receives the length of the ship value in meters.

- WX GUI Text Box ID: “width” – Receives the width of the ship value in meters.
- WX GUI Notebook ID: “notebook” – Creates an interface to accommodate all the text boxes and choosers.
- WX GUI Text Box ID: “Long” – Receives the longitude value for the position of the ship
- WX GUI Text Box ID: “Lat” – Receives the latitude value for the position of the ship
- WX GUI Text Box ID: “COG” – Receives the course over ground value in degrees.
- WX GUI Text Box ID: “cs” – Receives the callsign of the ship.
- WX GUI Text Box ID: “name” – Receives the name of the ship.
- WX GUI Chooser ID: “channel_select” – Allows the user to choose the AIS channel and which message does he wants to send. The available options are: Dynamic - CHA, Dynamic - CHB, Dynamic - Both, Static – A and Static – B.
- XMLRPC Client – Same function as in 3.2.1.1.

3.2.6.2 Server



Options ID: top_block Generate Options: No GUI Run Options: Run to Completion

Variable ID: samp_rate Value: 326.531k

Variable ID: bit_rate Value: 9.6k

XMLRPC Server Address: 192.168.1.103 Port: 12.345k

Variable ID: Dynamic Value: 00000100...00111110011

Variable ID: PartA Value: 01100000...00000000000

Variable ID: mmsi Value: 123.123M

Variable ID: Long Value: -9.6

Variable ID: name Value: TEST

Variable ID: NAVSTAT Value: 15

Variable ID: Lat Value: 38.5

Variable ID: cs Value: ABCD

Variable ID: SOG Value: 10

Variable ID: COG Value: 20

Variable ID: channel_select Value: 0

Variable ID: PartB Value: 01100000...00101000000

Variable ID: cslenght Value: ABCD@@@

Variable ID: letter0 Value: T

Variable ID: letter1 Value: E

Variable ID: letter2 Value: S

Variable ID: letter3 Value: T

Variable ID: letter4 Value: @

Variable ID: letter5 Value: @

Variable ID: letter6 Value: @

Variable ID: letter7 Value: @

Variable ID: letter8 Value: @

Variable ID: letter9 Value: @

Variable ID: tipe Value: 50

Variable ID: letter10 Value: @

Variable ID: letter11 Value: @

Variable ID: letter12 Value: @

Variable ID: letter13 Value: @

Variable ID: letter14 Value: @

Variable ID: lenght Value: 90

Variable ID: letter15 Value: @

Variable ID: letter16 Value: @

Variable ID: letter17 Value: @

Variable ID: letter18 Value: @

Variable ID: cs0 Value: A

Variable ID: width Value: 10

Variable ID: cs1 Value: B

Variable ID: cs2 Value: C

Variable ID: cs3 Value: D

Variable ID: cs4 Value: @

Variable ID: cs5 Value: @

Variable ID: cs6 Value: @

Variable ID: amp Value: 900m

Variable ID: vocabolary Value: @ABCDEFG...56789:;<=>?

Variable ID: namebin Value: 01010000...00000000000

Variable ID: nametest Value: TEST@@@@@@@@@@@@@@@

Variable ID: csbin Value: 00000100...00000000000

GMSK Mod Samples/Symbol: 34 BT: 400m

AIS Frame Builder Sentence: 011000...0000000000 Repeat: Yes Enable_NRZI_Conversion: Yes

Signal Source Sample Rate: 326.531k Waveform: Sine Frequency: -25k Amplitude: 1 Offset: 0

Multiply Const Constant: 900m

AIS Frame Builder Sentence: 000001...0111110011 Repeat: Yes Enable_NRZI_Conversion: Yes

GMSK Mod Samples/Symbol: 34 BT: 400m

Signal Source Sample Rate: 326.531k Waveform: Sine Frequency: -25k Amplitude: 1 Offset: 0

Multiply Const Constant: 900m

Multiply Const Constant: 450m

Selector Input Index: 0 Output Index: 0

osmocom Sink Device Arguments: ha...935ccf Sample Rate (sps): 326.531k Ch0: Frequency (Hz): 162M Ch0: Freq. Corr. (ppm): 0 Ch0: RF Gain (dB): 0 Ch0: IF Gain (dB): 0 Ch0: BB Gain (dB): 0

Multiply Const Constant: 450m

AIS Frame Builder Sentence: 000001...0111110011 Repeat: Yes Enable_NRZI_Conversion: Yes

GMSK Mod Samples/Symbol: 34 BT: 400m

Signal Source Sample Rate: 326.531k Waveform: Sine Frequency: 25k Amplitude: 1 Offset: 0

Multiply Const Constant: 900m

Signal Source Sample Rate: 326.531k Waveform: Sine Frequency: -25k Amplitude: 1 Offset: 0

AIS Frame Builder Sentence: 011000...0101000000 Repeat: Yes Enable_NRZI_Conversion: Yes

GMSK Mod Samples/Symbol: 34 BT: 400m

Multiply Const Constant: 900m

Figure 19 - AIS Spoofer Server Block Diagram

The used blocks were:

- Variables ID: “mmsi”, “NAVSTAT”, “SOG”, “Long”, “Lat”, “COG”, “name”, “cs”, “channel_select”, “tipe”, “length” and “width” –Variables that receive the values from the client through the XMLRPC Server block.
- Variable ID: “Dynamic” – Builds the Dynamic AIS message (type 1), making it ready to introduce in the AIS Frame Builder. The message is built according to Table 9:

Table 9 - AIS Dynamic Message Sentence

Nº Bits	Description	Message Part
6	Message Type	000001

2	Repeat Indicator	00
30	MMSI	"{0:b}".format(mmsi).rjust(30,'0')
4	Navigation Status	"{0:b}".format(NAVSTAT).rjust(4,'0')
8	Rate of Turn	10000000
10	SOG	"{0:b}".format(SOG*10).rjust(10,'0')
1	Position Accuracy	1
28	Longitude	'{0:b}'.format(int(round(Long*600000))& 0b1111111111111111111111111111).rjust(28,'0')
27	Latitude	'{0:b}'.format(int(round(Lat*600000))& 0b111111111111111111111111111).rjust(27,'0')
12	COG	"{0:b}".format(COG).rjust(12,'0')
9	True Heading	111111111
6	Time Stamp	101000
2	Manoeuvre Indicator	00
3	Spare	000
1	RAIM Flag	1
19	Radio Status	0010100000111110011

- Variable ID: "Part A" - Builds the Static AIS message (type 24), making it ready to introduce in the AIS Frame Builder. The type 24 AIS message is divided in two parts: A and B. The part A message is built according to Table 10:

Table 10 - AIS Static Part A Message Sentence

Nº Bits	Description	Message Part
6	Message Type	011000
2	Repeat Indicator	00

30	MMSI	"{0:b}".format(mmsi).rjust(30,'0')
2	Part Number	00
120	Vessel Name	namebin+('0'*(156-6-2-30-2-len(name)))

- Variable ID: “Part B” – Part B of the static type 24 AIS Message. The part B message is built according to Table 11. The dimensions present on the static type 24 AIS message are the distances from the AIS antenna. Here, it’s considered that the antenna is in the middle of the ship.

Table 11 - AIS Static Part B Message Sentence

Nº Bits	Description	Message Part
6	Message Type	011000
2	Repeat Indicator	00
30	MMSI	"{0:b}".format(mmsi).rjust(30,'0')
2	Part Number	01
8	Ship Type	'{0:b}'.format(tipe).rjust(8,'0')
18	Vendor ID	000000000000000000000000000000000000000000
42	Callsign	csbin (Variable)
9	Dimension to Bow	'{0:b}'.format(lenght/2).rjust(9,'0')
9	Dimension to Stern	'{0:b}'.format(lenght/2).rjust(9,'0')
6	Dimension to Port	'{0:b}'.format(width/2).rjust(6,'0')
6	Dimension to Starboard[1]	'{0:b}'.format(width/2).rjust(6,'0')
6	Spare	000000

- Variable ID: “Vocabolary” – List of 6-bit ASCII characters used in AIS sentences.
- Variable ID: “nametest” – Creates a 20-character sentence containing the name of the ship and, if the name isn’t 20 characters long, it replaces the empty spaces with the “@” character. The “@” is the first character in the 6-bit ASCII

vocabulary (000000). That’s why it’s the character replacing empty spaces. The code used is: *name+"@"*(19-len(name))*.

- Variable ID: “letter(number)” – Variable that reads the character on the “nametest” variable. There is one “letter” variable for each character. The code used is: *nametest[number]*.
- Variable ID: “namebin” – It replaces each character in the “letter” variables for its 6-bit ASCII code equivalent. The code used is: *'{0:b}'.format(vocabolary.find(letter0)).rjust(6, '0') +* *'{0:b}'.format(vocabolary.find(letter1)).rjust(6, '0') + …*
- Variable ID: “cslenght” – Same as the “nametest” variable but applied to the callsign.
- Variable ID: “csbin” – Same as the “namebin” variable but applied to the callsign.
- Variable ID: “amp” – Variable used to define the amplitude of the signals.
- AIS Frame Builder – Block that transforms the sentences from the Dynamic, Part A and Part B variables into an AIS Frame.
- GMSK Mod – Block that modulates the AIS Frame using Gaussian Minimum Shift Key modulation.
- Selector – Selects what signal to send to the “osmocom Sink” block.
- Signal Source, osmocom Sink, Multiply and Add – Same function as in 3.2.1.2 and 3.2.2.2.

3.2.7 Remote Control

To implement the EW package and running the programs in an autonomous vehicle, it was necessary to create a way of controlling them remotely. In order to do that, two external programs had to be created. These programs (programmed in Python), the “Client” and the “Server” are in Appendix A and B, with the explained script and steps.

The “Client” is the program that runs on the GCS and is responsible for sending the messages and commands to the UxS and to open the interface programs on the GCS. It allows the user to input the IP address and port for communicating with the UxS. After introducing these parameters, the program asks the user what program does he which to initiate, being the available programs the AIS Jammer, the AIS Spoofer, the Free Jammer, the GNSS Jammer, the WiFi Jammer and the WiFi Jammer All Band.

The “Server” is the program that runs in the UxS and it deals with the commands received from the “Client” and starts the previously mentioned programs on the Raspberry Pi. Also, the “Server” sends messages to the GCS to provide confirmation and status information to the user.

With this, the system ends up with the following architecture (Figure 20 and 21):

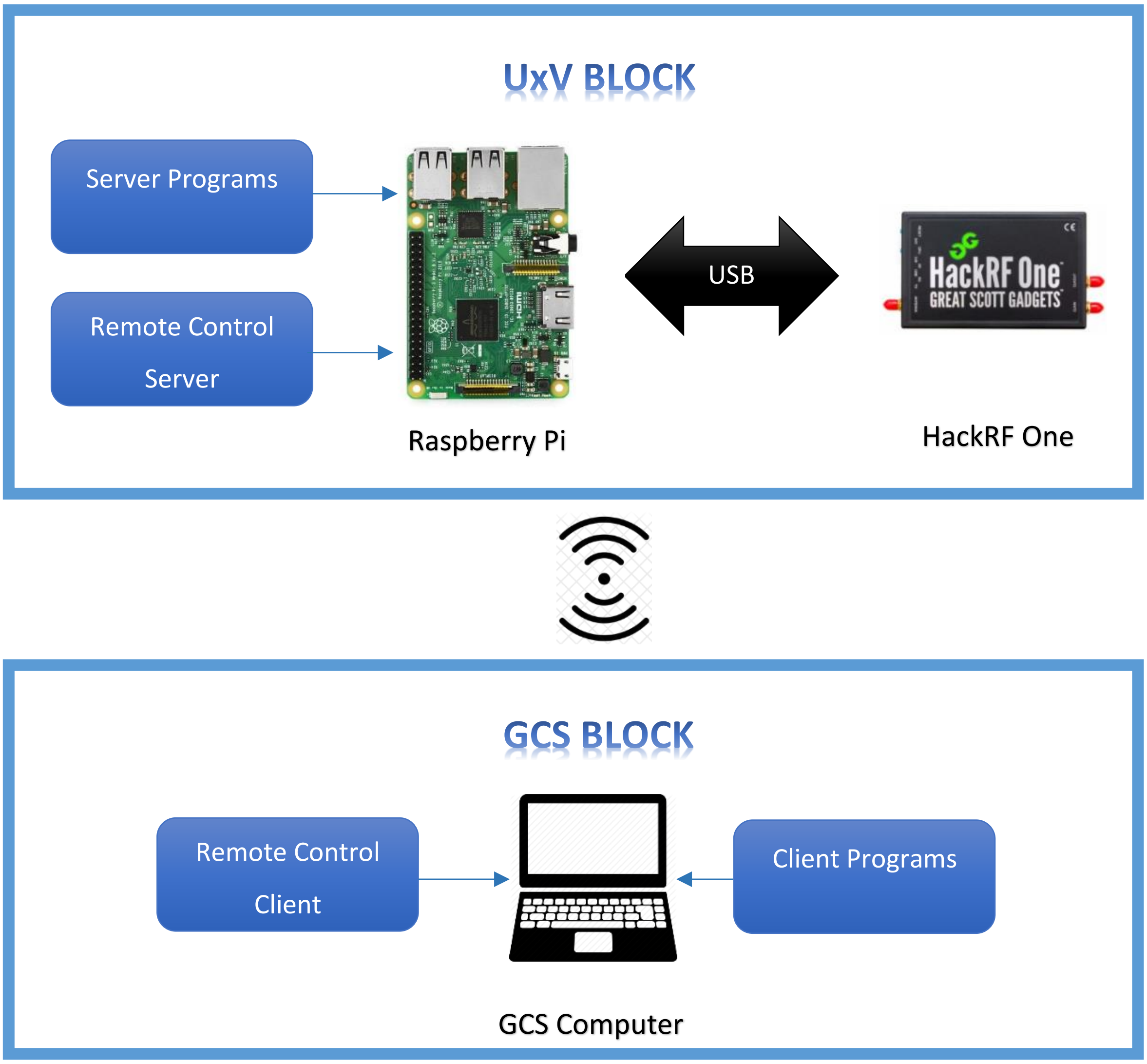


Figure 20 - EW Package Physical Architecture

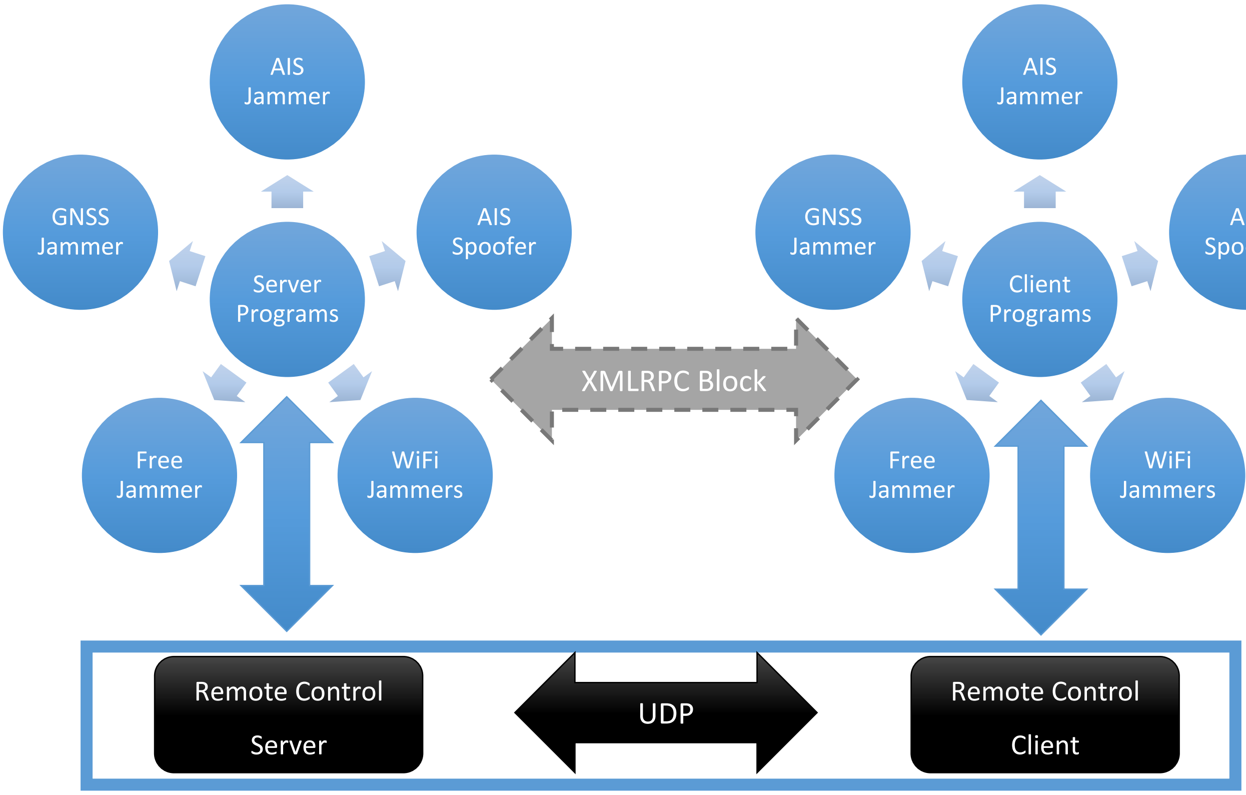


Figure 21 - EW Package Functional Architecture

4 Experiments and Results

In order to prove the effectiveness and the applicability of the Jammer, some tests were made. In this chapter, those tests are presented.

4.1 Portuguese Naval Academy Test

On the 14th of January 2019, a test was made at the Portuguese Naval Academy. The objective was to test if the WiFi Jammer was effective against an UAV commercial off the shelf (COTS). The UAV used for this test was a “Parrot Bebop 2” which is controlled via an app on a mobile phone along with the “Parrot Skycontroller 2” for increased range with a transmission power of 24dBm and an antenna gain of 5dBi. The UAV’s control signal and video link frequency are in the 2.4 GHz band. Specifically, for this test, WiFi channel 11 (2.462 GHz) was chosen in order to reduce the interference to other working networks, after an analysis has been made to the surrounding environment.

On the Jammer side, the used program was the WiFi Jammer, running on a laptop along with the HackRF, a 1W WiFi signal amplifier and a 7dBi WiFi antenna. Once the used antenna was omnidirectional, a home-made reflector was made in order to increase its directivity. With a spectrum analyser, a test was made in order to detect the power that the WiFi amplifier was really emmiting. The registered value 42dbm above the noise level. The power amplifier was 1 meter away from the antenna of the spectrum analyser. In Figure 22, a screenshot of the spectrum analyser is shown:

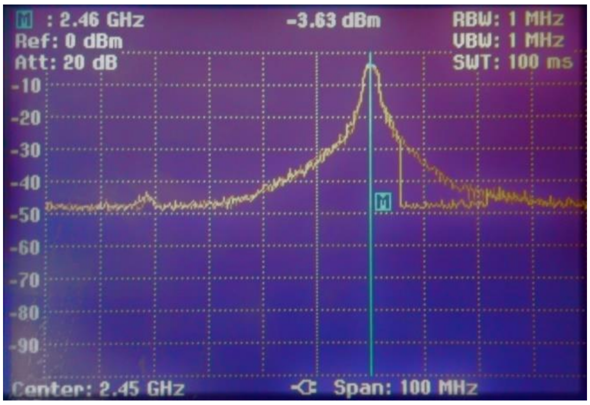


Figure 22 - Spectrum Analyser Screenshot

According to the Wireless-Nets [47], the minimum signal to noise ratio (SNR) for a wireless communication to be able to be established and to be stable is 10 dB. With this, in order to jam the signal between the remote controller and the UAV if they're 1 meter apart (SNR=53dB) [48], the jammer to signal ratio must be approximately -43dB. So, in order to determine the maximum theoretical jamming distance, the Friis equation is applied:

$$\frac{J}{S} = \frac{\frac{P_J G_J G_R \lambda^2}{(4\pi d_J)^2}}{\frac{P_T G_T G_R \lambda^2}{(4\pi d_S)^2}} \tag{1}$$

Simplifying and converting to dB:

$$\frac{J}{S} = P_J + G_J - P_T - G_T + 20\log d_S - 20\log d_J \tag{2}$$

Being:

- J the jammer signal power at intended receiver (dB)
- S the transmitter signal power at intended receiver (dB)
- P_J the jammer signal output power (dBW)
- P_T the transmitter output power (dBW)
- G_J the jammer antenna gain (dBi)
- G_T the transmitter antenna gain (dBi)
- d_J the distance from the jammer to the receiver (m)
- d_S the distance from the transmitter to the receiver (m)

Being d_J the desired value, the final equation is:

$$d_J = 10^{\frac{P_J + G_J - P_T - G_T + 20\log(d_S) - \frac{J}{S}}{20}} \tag{3}$$

After substituting the values, the obtained theoretical value for the maximum range of the jammer was 376 meters.

Figure 23 shows the Jammer set:

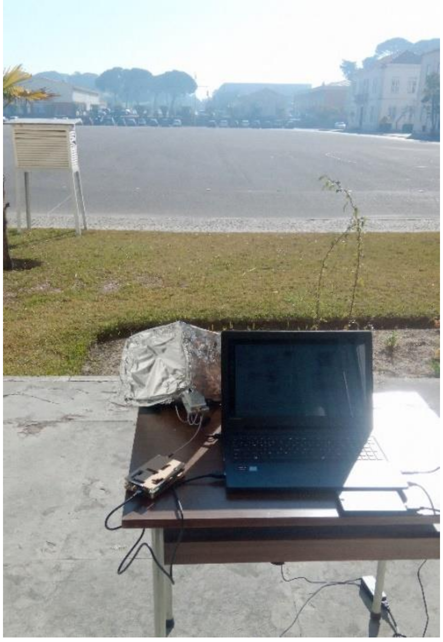


Figure 23 - Jammer set on the Naval Academy’s Test

To do the test, the UAV and controller were both with the operator within a maximum distance of 1 meter. After turning on the Jammer, the operator started to move away from the jammer in a straight line. The stopping point for the operator would be either when the Jammer stopped being effective or when he reached the limit of the Academy’s perimeter.

Figure 24 shows the points where the Jammer was and the point where the UAV and the remote controller was:

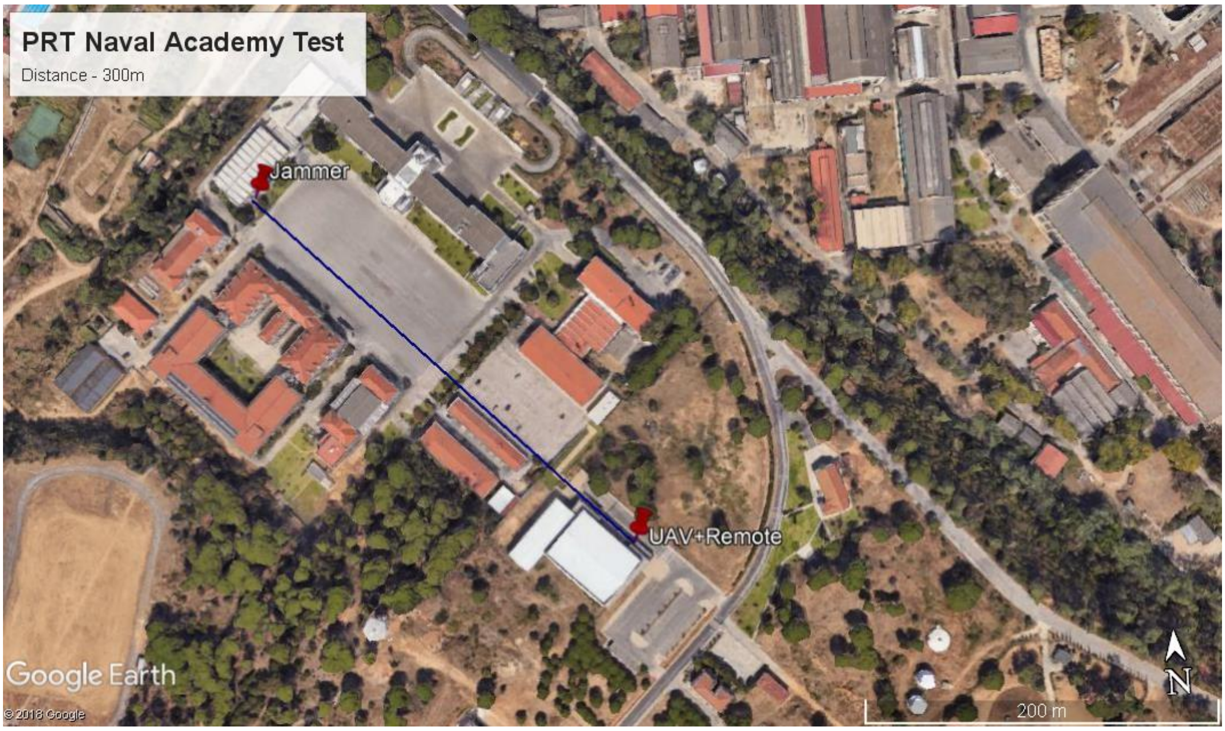


Figure 24 - Naval Academy's Jamming Test

The Jammer was effective until the operator reached the Academy's perimeter, totalling an effective distance of 300 meters, with a free-space pathloss of 89.20dB, as described in the following equation, being d the distance in kilometres and f the frequency in MHz:

$$FSPL(dB) = 32.5 + 20\log d + 20\log f \qquad (2)$$

Between the jammer and the UAS operator there were some obstacles, such as cars, trees and fences.

4.2 Portuguese Marines Base Test

Another test has been made during an EW exercise against UAVs COTS in the 17th of January 2019. This time, the test was done at the Portuguese Marines Base in the Lisbon Naval Base. The purpose of this exercise was to test the available capacities of the Portuguese Marines EW Unit against UAVs, using several techniques and equipment available on a portable container designed to be easily deployed and to assist amphibious operations.

To test the Wi-Fi Jammer, all the parameters of the exercise were similar to the ones in the Naval Academy's Test. Figures 25 and 26 show the set of the Jammer and the points where the operator and the Jammer were, respectively:

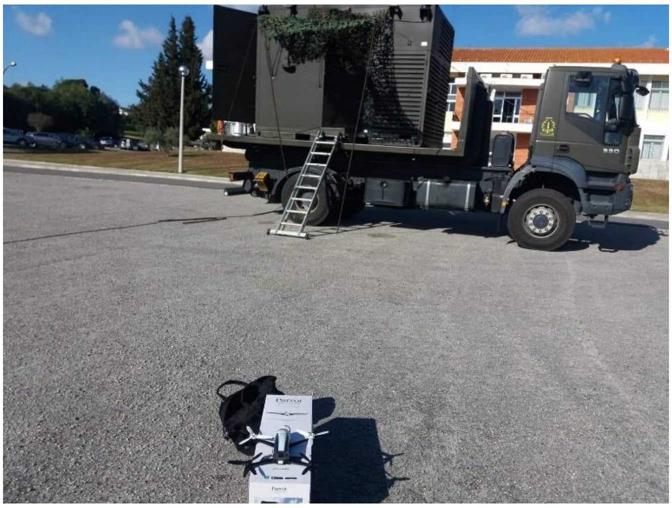


Figure 25 - Portuguese Marines container and Parrot Bepop 2

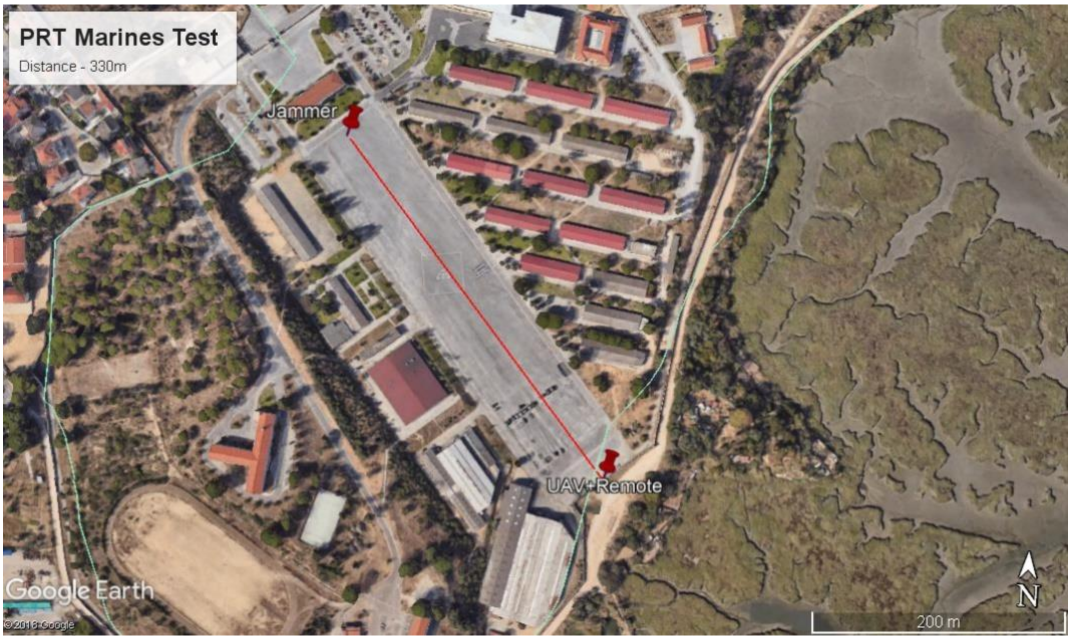


Figure 26 - Portuguese Marines Base Jamming Test

In this case, the jamming was also effective in the measured distance of 330 meters with a free-space pathloss of 90.70dB.

4.3 Lisbon Naval Base Test

On the 12th of February 2019, a test was made at the Lisbon Naval Base in order to determine how effective were the Portuguese Navy's defences against UAS COTS and to determine if the Wi-Fi Jammer was effective against these threats on a more realistic scenario. The exercise's report in presented on Appendix C.

The exercise was composed by a series of approaches with UAS COTS to two ships (NRP António Enes and NRP Vasco da Gama) in order to verify the crew's procedures to the presence of unauthorized UAS flying on the Naval Base perimeter. To do this, several approaches were made at different altitudes and speeds with two types of UAVs: a fixed-wing and a rotary-wing. After this, a jamming test was made to test the performance of this defensive measure against an UAS approaching at its maximum velocity.

The flight plan of the fixed-wing UAV during the exercise is presented on Figure 27:

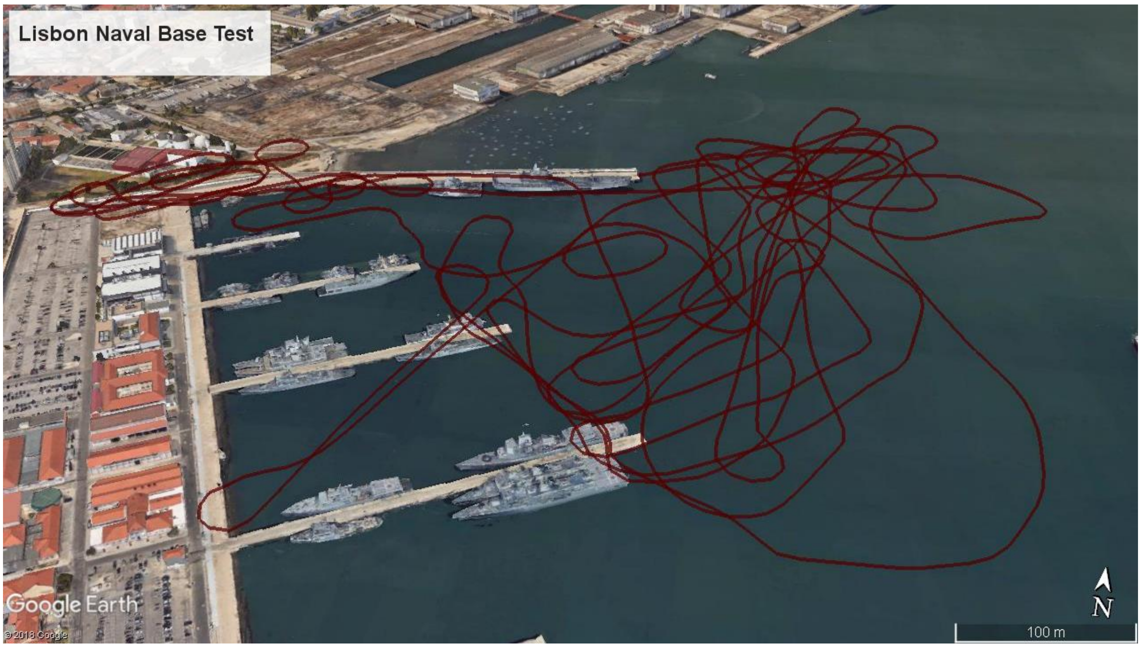


Figure 27 - Naval Base Test Fixed-Wing UAV Flight Plan

The overall results of the test regarding the use of the Wi-Fi Jammer were good. The jammer was capable of making the UAV to stop on a stationary flight, heavily reducing the threat that it posed.

4.4 NRP Álvares Cabral Test

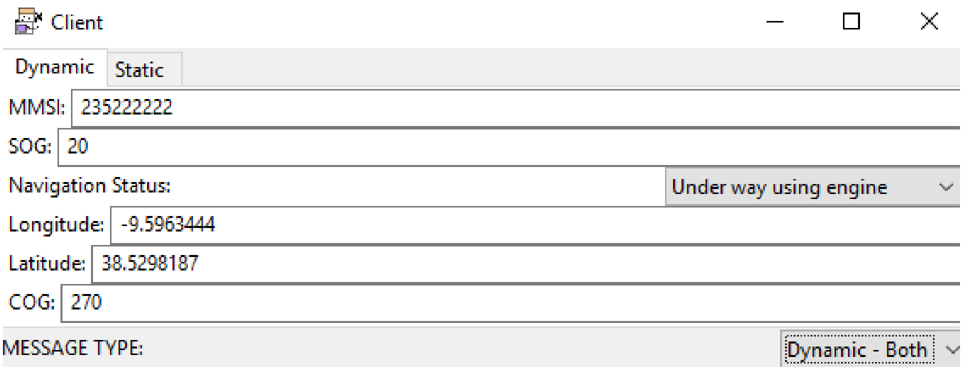
On the 3rd of February 2019, a test was made on board the Portuguese Navy ship NRP Álvares Cabral docked at the Lisbon Naval Base, in order to evaluate if the AIS Spoofer could create fake contacts on an actual AIS Transponder.

Because there were other ships in the area, the output power used on the AIS Spoofer was kept to its minimum to not interfere with other operations in the Base. To ensure that the AIS antenna of the ship received the emitted signal, the EW Package was placed within a few meters of the antenna.

The tested modes were the “Dynamic – Both”, “Static – A” and “Static – B”. For the “Dynamic – Both” mode the configuration was the following:

- MMSI: 235222222;
- SOG: 20 KTS;
- Navigation Status: Under way using engine;
- Longitude: -9.5963444;
- Latitude: 38.5298187;
- COG: 270.

These parameters were set to the AIS Spoofer as shown in Figure 28.



MMSI: 235222222
SOG: 20
Navigation Status: Under way using engine
Longitude: -9.5963444
Latitude: 38.5298187
COG: 270
MESSAGE TYPE: Dynamic - Both

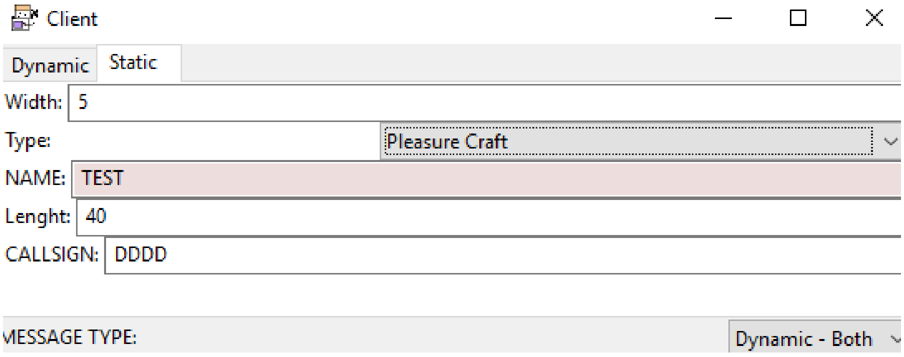
Figure 28 - "Dynamic - Both" Test

For the “Static – A” and “Static – B” mode tests, the configuration was the following:

- Width: 5 m;
- Type: Pleasure Craft;

- Name: TEST;
- Length: 40 m;
- Callsign: DDDD

These parameters were set to the AIS Spoofer as shown in Figure 29:



Width: 5

Type: Pleasure Craft

NAME: TEST

Lenght: 40

CALLSIGN: DDDD

MESSAGE TYPE: Dynamic - Both

Figure 29 – “Static – A” and “Static – B” Test

The NRP Álvares Cabral’s AIS transponder was able to receive a valid AIS message on all modes, with all the tests considered to be concluded with success.

5 Conclusion

All the tests that were made, proved to be successful. Even though only the Wi-Fi Jammer had medium range tests, all the other jammers were tested on a spectrum analyzer with lower output power values, matching the expected results.

The proposed hypothesis to the research question: **“By using an SDR transceiver and a Raspberry Pi computer, it is possible to build a device with jamming and spoofing capabilities on various communication systems, capable of being implemented on an UxS.”**, has been proved.

The jamming and spoofing capabilities mentioned above, constituted by an AIS Jammer, a Free Jammer, a Wi-Fi Jammer, a Wi-Fi Jammer in all band, a GNSS Jammer and an AIS Spoofer have been achieved using the proposed components (an SDR and the Raspberry Pi computer). The HackRF One as the chosen SDR had a good performance running the programs being its biggest limitation the 20MHz available bandwidth which, in the Wi-Fi jammer All Band, lead to the need of further development on the program, in order to jam all the Wi-Fi band (approximately 100MHz). The Raspberry Pi also dealt well with the programs however, the small size of this computer comes with a prize: the processing capacity is smaller than a normal size computer. This reduced processing capacity lead to several necessary adjustments to the programs such as the enlargement of the transition width of the low-pass filters resulting in a smaller efficiency on the transmitted power and the limitation on the sample rate resulting in emitted signals with lower bandwidths. But, even with these limitations on the components, the overall result proves that the hypothesis to the research question is possible.

The importance of improving EW capacities was why this EW package was developed. This importance is stated by NATO, which recognizes its “limited capability for EA”, considering that this capability “is expected to grow as operational lessons have identified the need” [14]. Controlling the EM spectrum has a high impact on the success of military operations, especially if the units are highly dependent on wireless communication systems such as the ones developed in this EW package. Being able to execute deception operations or deny those systems to the adversary is of great importance. Also, embarking jammers on autonomous vehicles has a high potential because one of the major problems with jamming is that, when transmitting, the jamming unit/source reveals its position, potentializing situations of grave danger. By separating the jammer from the operators, this danger is heavily reduced.

5.1 Future Work

The propositions for future work, based on the developed work in this thesis, are the development of a detector for irregular EM emissions on frequencies of strategic interest, using an SDR and GNU Radio, in order to improve ES capabilities and to complement the use of the EW package developed in this thesis.

Also, the development of new programs, adding capacities and expanding EA capabilities to new communication systems would bring benefits for the final user of the EW package

Finally, the development of another communications' protocol between the embarked EW package and the GCS, possibly considering frequency hopping or encoded communications, would create a more robust and functional communications method.

6 Bibliography

[1] M. M. Marques, “Reference Model for Interoperability of Autonomous Systems,” Universidade Nova de Lisboa, 2018.

[2] S. MahmoudZadeh, D. M. W. Powers, and R. Bairam Zadeh, *Autonomy and Unmanned Vehicles*. Springer, 2019.

[3] NATO, *UAS Tactical Pocket Guide*. NATO, 2014.

[4] Joint Chief of Staff, “Joint Publication 3-13.1: Electronic Warfare,” 2007.

[5] K. H. Hicks, A. Metrick, L. S. Samp, and K. Weinberger, *Undersea Warfare in Northern Europe*. Rowman & Littlefield, 2016.

[6] P. Mcleary, “Russians Tried to Jam NATO Exercise; Swedes Say They’ve Seen This Before,” 2018. [Online]. Available: https://breakingdefense.com/2018/11/russians-tried-to-jam-nato-exercise-swedes-say-theyve-seen-this-before/. [Accessed: 19-Aug-2019].

[7] S. Crarford and L. Stucki, “Peer review and the changing research record,” *J. Am. Soc. Inf. Sci.*, vol. 41, pp. 223--228, 1990.

[8] R. Poisel, *Introduction to Communication Electronic Warfare Systems 2nd Ed.* Norwood, MA: Artech House, Inc, 2002.

[9] NATO RESEARCH AND TECHNOLOGY ORGANIZATION, “Electronic Warfare Test and Evaluation,” 2000.

[10] D. C. Schleher, *Electronic Warfare in the Information Age*. 1999.

[11] D. Adamy, *EW 101 - A First Course in Electronic Warfare*. 2001.

[12] W. Vigder, “Electronic Protection Measures In Modern Anti-Ship Missiles,” Ottawa, 2013.

[13] D. Adamy, *EW 102: A Second Course in Electronic Warfare*. Horizon House Publication, Inc., 2004.

[14] NATO, "Allied Joint Doctrine for Electronic Warfare - AJP-3.6 Edition B Version 1." 2012.

[15] R. Poisel, *Modern Communications Jamming Principles and Techniques*. Artech House, Inc, 2011.

[16] E. Ström, T. Ottosson, and A. Svensson, "An Introduction to Spread Spectrum Systems," Göteborg, 2002.

[17] J. K. Holmes, *Spread Spectrum Systems for GNSS and Wireless Communications*. Norwood, MA: Artech House, Inc, 2007.

[18] V. P. Ipatov, *Spread Spectrum and CDMA*. Chichester, UK: John Wiley & Sons, Ltd, 2005.

[19] J. C. Whitaker, *The Electronics Handbook*. CRC press, 2005.

[20] Joint Chiefs of Staff, "Joint Publication 1-02 - Department of Defense Dictionary of Military and Associated Terms," 2001.

[21] L. Seidl R., "Multi-Tone Jammer," 1982.

[22] K. Grover, A. Lim, and Q. Yang, "Jamming and anti-jamming techniques in wireless networks: a survey," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 17, no. 4, p. 197, 2014.

[23] I. M. Organization, "SOLAS - International Convention for the Safety of Life at Sea." pp. 1–910, 1974.

[24] A. Harati-Mokhtari, A. Wall, P. Brooks, and J. Wang, "Automatic Identification System (AIS): Data Reliability and Human Error Implications," *J. Navig.*, vol. 60, no. 3, pp. 373–389, 2007.

[25] I. M. Organization, "Resolution Maritime Safety Committee 74," 1998.

[26] M. Traffic, "What is the Automatic Identification System." [Online]. Available: https://help.marinetraffic.com/hc/en-us/articles/204581828-What-is-the-Automatic-Identification-System-AIS-. [Accessed: 16-Jun-2019].

[27] M. Balduzzi, A. Pasta, and K. Wilhoit, "A Security Evaluation of AIS Automated Identification System," in *Proceedings of the 30th Annual Computer Security*

Applications Conference, 2014, pp. 436–445.

[28] I. T. U. ITU, "ARTICLE 1 - Terms and Definitions," *1.15*, pp. 1–19, 2009.

[29] E. C. C. (ECC), "The European Table Of Frequency Allocations And Applications In The Frequency Range 8 . 3 Khz To 3000 Ghz (ECA Table)," *Eur. Conf. Post. Telecommun. Adm.*, no. October, pp. 1–262, 2015.

[30] P. B. Khannur, "A CMOS power amplifier with power control and T/R switch for 2.45-GHz Bluetooth/ISM band applications," in *IEEE Radio Frequency Integrated Circuits (RFIC) Symposium, 2003*, 2003, pp. 145–148.

[31] P. M. Cunha, "Processamento de Dados GNSS em Modo Cinemático," Escola Naval, 2016.

[32] T. M. L. Brown, S. A. Mccabe, and C. Wellford, "Global Positioning System (GPS) Technology for Community Supervision: Lessons Learned," 2007.

[33] J. S. Warner and R. G. Johnston, "GPS Spoofing Countermeasures," *Homel. Secur. J.*, vol. 25.2, pp. 19–27, 2003.

[34] B. G. Ventorim and W. R. D. Poz, "Performance evaluation of GPS and GLONASS systems, combined and individually, in precise point positioning," *BCG - Bol. Ciências Geodésicas*, vol. 22, pp. 264–281, 2016.

[35] J. Pérez Bartolomé, X. Maufroid, I. Fernandez-Hernandez, J. A. López-Salcedo, and G. Seco-Granados, "Overview of Galileo System," J. Nurmi, S. Lohan, H. Kursaainen, and S. Sand, Eds. Springer Ltd, 2014.

[36] T. J. Willink, "SDR and Cognitive Radio for Military Applications," Ottawa, 2007.

[37] I. T. U. ITU, "Definitions of Software Defined Radio (SDR) and Cognitive Radio System (CRS)," Geneva, 2009.

[38] F. C. Commission, O. of E. and Technology, and L. Division, "SOFTWARE DEFINED RADIO APPLICATION GUIDE," 2014.

[39] M. Ossmann, "HackRF One," *Great Scott Gadgets*, 2017. [Online]. Available: https://greatscottgadgets.com/hackrf/one/. [Accessed: 16-Jun-2019].

[40] T. Senator, P. Leahy, and D. Investigation, “HackRF One.” Champlain College, 2017.

[41] I. The GNU Radio Foundation, “GNU Radio,” 2019. [Online]. Available: https://www.gnuradio.org/about/. [Accessed: 16-Jan-2019].

[42] R. P. FOUNDATION, “Raspberry Pi,” 2019. [Online]. Available: https://www.raspberrypi.org/. [Accessed: 26-May-2019].

[43] V. Vujovic and M. Maksimovic, “Raspberry Pi as a Wireless Sensor node: Performances and constraints,” in *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2014, pp. 1013–1018.

[44] E. S. Raymond, “AIVDM/AIVDO protocol decoding.” [Online]. Available: http://catb.org/gpsd/AIVDM.html. [Accessed: 14-Mar-2019].

[45] N. C. of Excellence, “AIS MESSAGES.” [Online]. Available: https://www.navcen.uscg.gov/?pageName=AISMessages. [Accessed: 16-Jun-2019].

[46] M. Balduzzi, “AIS Frame Builder Block.” [Online]. Available: https://github.com/trendmicro/ais. [Accessed: 19-Mar-2019].

[47] J. Geier, “How to: Define Minimum SNR Values for Signal Coverage,” 2013. [Online]. Available: http://www.wireless-nets.com/resources/tutorials/define_SNR_values.html. [Accessed: 15-Jul-2019].

[48] F. C. Commission, “APPLICATION CERTIFICATION FCC Part 15C On Behalf of Parrot Drone SAS,” 2017.

7 Appendixes

7.1 Appendix A - Client

```
#!/usr/bin/env python


import os
import socket
from socket import *
import sys
import xmlrpclib
import subprocess
from subprocess import check_output
import time         #Dependecies import

def main():
       host=raw_input("IP: ")     #Input for the IP
       port=raw_input("PORT: ")   #Input for the Port
       port=int(port)
       addr = (host, port)
       UDPSock = socket(AF_INET, SOCK_DGRAM)    #Creation of the UDP socket

       print "LIST OF AVAILABLE PROGRAMS: \n TEST \n AIS JAMMER \n AIS SPOOFER \n 
FREE JAMMER \n GNSS JAMMER \n WIFI JAMMER \n WIFI JAMMER ALL BAND \n PEAK FINDER 
\n  EXIT \n"  #List of available programs
#Program Choosing


       inp=raw_input("Choose Program: ")

       if (inp == "TEST") or (inp == "Test") or (inp == "test"):    #Test Program

              subprocess.Popen(["/home/diogo/Desktop/Test/top_block.py"])

              time.sleep(1)

              def test():

                     try:
                            s = xmlrpclib.Server('http://localhost:8080')
                     except socket.error:
                            print 'Failed to create socket'
                            sys.exit()

                     while(1) :
                            msg = raw_input('Enter message to send :')
                            msg=int(msg)
                            try :
                                   #Set the whole string
                                   s.set_freq(msg)

                            except socket.error, msg:
                                   print 'Error Code : ' + str(msg[0]) + ' Message 
' + msg[1]

                                   #sys.exit()
                            test()
              test()
       #AIS SPOOFER
       elif (inp == "ais spoofer") or (inp == "AIS SPOOFER") or (inp == "Ais
Spoofer"):    #Possible inputs to open AIS Spoofer


       #p=subprocess.Popen(["/home/diogo/Desktop/GNU/AIS_SPOOFER/top_block.py"])

       pp=subprocess.Popen(["/home/diogo/Desktop/GNU/AIS_SPOOFER/client.py"])
       #Opens the client program
              data='aisspoofer'
              UDPSock.sendto(data, addr) #Sends the word on the variable data to
the Server in order to open the AIS Spoofer on the Raspberry Pi
              close=raw_input('Do you wish to stop Spoofing? [Y] : ')
       #Part of the cicle dedicated to close the program
              if (close=='Y') or (close=='y'):
                     #p.kill()
                     UDPSock.sendto("exitt", addr)
                     pp.kill()


       # AIS JAMMER
       elif (inp == "AIS JAMMER") or (inp == "Ais Jammer") or (inp == "ais
jammer"):


       #p=subprocess.Popen(["/home/diogo/Desktop/GNU/AIS_JAMMER/top_block.py"])

       pp=subprocess.Popen(["/home/diogo/Desktop/GNU/AIS_JAMMER/client.py"])
              data='aisjammer'
              UDPSock.sendto(data, addr)
              close=raw_input('Do you wish to stop Jamming? [Y] : ')
              if (close=='Y') or (close=='y'):
                     #p.kill()
                     UDPSock.sendto("exitt", addr)
                     pp.kill()
                     main()

       # FREE JAMMER
       elif (inp == "FREE JAMMER") or (inp == "Free Jammer") or (inp == "free
jammer"):


       #p=subprocess.Popen(["/home/diogo/Desktop/GNU/FREE_JAMMER/top_block.py"])

       pp=subprocess.Popen(["/home/diogo/Desktop/GNU/FREE_JAMMER/client.py"])
              data='freejammer'
              UDPSock.sendto(data, addr)
              close=raw_input('Do you wish to stop Jamming? [Y] : ')
              if (close=='Y') or (close=='y'):
                     #p.kill()
                     UDPSock.sendto("exitt", addr)
                     pp.kill()
                     main()

        # GNSS JAMMER
       elif (inp == "GNSS JAMMER") or (inp == "Gnss Jammer") or (inp == "gnss
jammer"):


       #p=subprocess.Popen(["/home/diogo/Desktop/GNU/GNSS_JAMMER/top_block.py"])
       pp=subprocess.Popen(["/home/diogo/Desktop/GNU/GNSS_JAMMER/client.py"])
              data='gnssjammer'
              UDPSock.sendto(data, addr)
              close=raw_input('Do you wish to stop Jamming? [Y] : ')
              if (close=='Y') or (close=='y'):
                     #p.kill()
                     UDPSock.sendto("exitt", addr)
                     pp.kill()
                     main()

        # WIFI JAMMER
       elif (inp == "WIFI JAMMER") or (inp == "Wifi Jammer") or (inp == "wifi
jammer"):

       #p=subprocess.Popen(["/home/diogo/Desktop/GNU/WIFI_JAMMER/top_block.py"])

       pp=subprocess.Popen(["/home/diogo/Desktop/GNU/WIFI_JAMMER/client.py"])
              data='wifijammer'
              UDPSock.sendto(data, addr)
              close=raw_input('Do you wish to stop Jamming? [Y] : ')
              if (close=='Y') or (close=='y'):
                     #p.kill()
                     UDPSock.sendto("exitt", addr)
                     pp.kill()
                     main()

       # WIFI JAMMER ALL BAND
       elif (inp == "wifi jammer all band") or (inp == "WIFI JAMMER ALL BAND") or
(inp == "Wifi Jammer All Band"):

              data='wifijammerallband'
              UDPSock.sendto(data, addr)
              close=raw_input('Do you wish to stop Jamming? [Y] : ')
              if (close=='Y') or (close=='y'):
                     UDPSock.sendto("exitt", addr)
                     main()

       # PEAK FINDER - This program is designed to find peaks on the EM spectrum
but it isn't contemplated on this work
       elif (inp == "PEAK FINDER") or (inp == "Peak Finder") or (inp == "peak
finder"):

       #p=subprocess.Popen(["/home/diogo/Desktop/GNU/PEAK_FINDER/top_block.py"])

       pp=subprocess.Popen(["/home/diogo/Desktop/GNU/PEAK_FINDER/client.py"])
              data='peakfinder'
              UDPSock.sendto(data, addr)
              close=raw_input('Do you wish to stop Jamming? [Y] : ')
              if (close=='Y') or (close=='y'):
                     #p.kill()
                     UDPSock.sendto("exitt", addr)
                     pp.kill()
                     main()

       # EXIT
       elif (inp == "EXIT") or (inp == "Exit") or (inp == "exit"):
              UDPSock.sendto("exitt", addr)     #Closes the program on the
Raspberry Pi
              sys.exit()
        else:
                print ("\n \n NOT AVAILABLE \n \n")      #If there is a typing
error, the program returns to the initial question
                time.sleep(1)
                main()

main()
```

7.2 Appendix B - Server

```
#!/usr/bin/env python

import os

from socket import *

import subprocess   #Dependencies import

host = ""           #Host that listens in any IP

port = 12345        #Port 12345

buf = 1024          #Buffer size: 1024

addr = (host, port) #Creation of an address containing the Host and the Port

UDPSock = socket(AF_INET, SOCK_DGRAM)           #Creation of the UDP Socket

UDPSock.bind(addr)                              #Definition of the UDP Socket's
address

print "Waiting to receive messages..."          #Message viewed by the User

while True:                                     #Cycle to keep the program
running

       (data, addr) = UDPSock.recvfrom(buf)     #Makes the program listen to
incoming messages on the address

       print "Received message: " + data        #Prints the data received

       if data == "aisspoofer":                              #AIS SPOOFER
       p=subprocess.Popen(["/home/pi/Desktop/GNU/AIS_SPOOFER/top_block.py"])
#Opens the AIS SPOOFER program

              (data, addr) = UDPSock.recvfrom(buf)

              if data == "exitt":

                     p.kill()      #Closes the program if received data is "exitt"

                     print "Received message: " + data

       if data == "aisjammer":    #AIS JAMMER
       p=subprocess.Popen(["/home/pi/Desktop/GNU/AIS_JAMMER/top_block.py"])

              (data, addr) = UDPSock.recvfrom(buf)

              if data == "exitt":

                     p.kill()

                        print "Received message: " + data

       if data == "freejammer":   #FREE JAMMER
       p=subprocess.Popen(["/home/pi/Desktop/GNU/FREE_JAMMER/top_block.py"])

              (data, addr) = UDPSock.recvfrom(buf)

              if data == "exitt":

                     p.kill()

                        print "Received message: " + data

       if data == "gnssjammer":   #GNSS JAMMER
       p=subprocess.Popen(["/home/pi/Desktop/GNU/GNSS_JAMMER/top_block.py"])

              (data, addr) = UDPSock.recvfrom(buf)

              if data == "exitt":

                     p.kill()

                         print "Received message: " + data

       if data == "wifijammer":   #WIFI JAMMER
       p=subprocess.Popen(["/home/pi/Desktop/GNU/WIFI_JAMMER/top_block.py"])

              (data, addr) = UDPSock.recvfrom(buf)

              if data == "exitt":

                     p.kill()

                         print "Received message: " + data

if data == "wifijammerallband": #WIFI JAMMER ALL BAND

p=subprocess.Popen(["/home/pi/Desktop/GNU/WIFI_JAMMER_ALLBAND/top_block.py"])

              (data, addr) = UDPSock.recvfrom(buf)

              if data == "exitt":

                     p.kill()

                         print "Received message: " + data

       if data == "peakfinder":   #PEAK FINDER

       p=subprocess.Popen(["/home/pi/Desktop/GNU/PEAK_FINDER/top_block.py"])

              (data, addr) = UDPSock.recvfrom(buf)

              if data == "exitt":

                     p.kill()

                     print "Received message: " + data

       if data == "exit":         #Shutdown of the program

              break

UDPSock.close()

os._exit(0)
```

7.3 Appendix C – Lisbon Naval Base Test Report

LISBON NAVAL BASE EXERCISE REPORT		
DATE: 12/02/2019	**PLACE:** Lisbon Naval Base	**PERIOD:** 1430 - 1540
NAVAL UNITS: -NRP VGAM; -NRP CORENES;		
UAV COTS: -PARROT DISCO FPV (UAV1); -PARROT BEBOP 2 FPV (UAV2);	**METEOROLOGY:** -Clear skies; -Wind 4 knots;	

1442 – Beginning of the Exercise; UAV1 take-off.
1443 – UAV1 sighted by the Observing Team aboard NRP VGAM; Duty Officer of NRP Bérrio contacts Lisbon Naval Base's Duty Officer reporting that two ASPOF are conducting an exercise with UAVs on pier 8.
1444 – UAV1 initiates 120m approach.
1445 – UAV1 sighted by NRP VGAM's Duty Officer.
1446 – Attempt (Failed) of NRP VGAM's Duty Officer to report UAV1 sighting to the COMNAV (by radio and telephone).
1447 – UAV1 begins 100m approach; NRP VGAM's Duty Officer contacts Lisbon Naval Base's Duty Officer (by telephone) and reports the sighting of UAV1. Patrol sailor loses sight of UAV1.
1449 – UAV1 begins 80m approach.
1451 – UAV1 does a 2nd 80m approach; NRP CORENES' Duty Officer sights UAV1 and reports it to the Naval Base Duty Officer; No other measures are taken.
1453 – UAV1 initiates 60m approach.
1454 – UAV1 is sighted by NRP VGAM's patrol sailor.
1455 – UAV1 initiates 40m approach; NRP CORENES' crew sights UAV1 but doesn't report it.
1456 – UAV1's operator loses control signal.
1458 – UAV1's operator recovers control signal.
1500 – UAV1 begins 20m approach.
1501 – NRP CORENES' crew sights UAV1 but doesn't report it.
1503 – UAV1 starts landing procedures.
1507 – UAV1 lands.
1517 – UAV2 take-off.
1518 – UAV2 is detected by NRP CORENES but there's no alarm.
1518 – UAV2 is detected by the Observing Team aboard NRP VGAM.
1521 – UAV2 lands to verify altimeter.
1522 – UAV2 take-off.
1528 – 1st high-speed approach with UAV2 to NRP VGAM; UAV2 loses signal due to Jamming defence measures and stays stationary.
1531 – 2nd high-speed approach with UAV2 to NRP VGAM; UAV2 loses signal due to Jamming defence measures and stays stationary.
1532 – UAV2 is sighted by NRP CORENES' Duty Officer.
1535 – End of exercise.

- Remarks:

a) Although the visibility conditions were excellent, the UAVs were hard to detect and track, even at 20m;
b) Apart from NRP Bérrio (whose Duty Officer sighted the UAV's operators at the beginning of the exercise) and the two ships participating on the exercise, no other ships reported the sighting of the UAVs. The two possible explanations for this may be: either the UAVs were not sighted, or they were sighted but not reported to the Naval Base's Duty Officer.
c) Apart from the communication between the Duty Officers, no other measures have been taken on board. After reporting the UAVs, no tracking was made in order to keep visual contact.